THE UNIVERSITY OF ALBERTA

SUBSTRUCTURE ANALYSIS OF PLANE FRAMES

by

© ALAA-ELDIN A. ELWI

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF CIVIL ENGINEERING

EDMONTON, ALBERTA

FALL, 1977

ABSTRACT

An equation solving package based on the skyline technique is developed for substructure analysis, as well as an assembly and a coordinate transformation scheme for the same purpose. Two plane frame substructure analysis programs, SISAPF and MUSAPF, are developed. The former is based on a single-level substructure scheme, and the latter is based on a multi-level substructure scheme.

A sample structure is partitioned in several ways. The core space parameters, and the CPU time requirements for the various stages of solution are discussed. A large saving in core space requirements, and a significant saving in CPU time are achieved.

ACKNOWLEDGEMENTS

The author wishes to express his deep appreciation to Dr. D.W. Murray, Professor of Civil Engineering, University of Alberta, for his supervision, patience, and support throughout the course of this study. Thanks are also due to Dr. S.H. Simmonds, and Dr. R. Bjorhovde, Department of Civil Engineering, and also to Dr. S. Cabay, Department of Computing Science, University of Alberta, for serving as members of the thesis committee.

Computer facilities were provided by the University of Alberta Computing Services at Edmonton.

The assistance of Nola Wismer who typed the manuscript is gratefully acknowledged.

ACKNOWLEDGMENTS

The author wishes to express his most sincere thanks to Dr. D.W. Murray, Professor of Civil Engineering, University of Alberta, for his supervision, guidance, and encouragement throughout the course of this study. Thanks are also due to Dr. [illegible], [illegible] and Dr. [illegible], Department of Civil Engineering, and Dr. [illegible], Department of Computing Science, University of Alberta, for serving as members of the thesis committee.

Computer facilities were provided by the University of Alberta Computing Services at [illegible].

The assistance of [illegible] who typed the manuscript is gratefully acknowledged.

TABLE OF CONTENTS

		Page
CHAPTER 1 - INTRODUCTION		1
1.1	Introductory Remarks	1
1.2	Full Release vs. Partial Release	3
1.3	Purpose and Scope	6
1.4	Outline of Contents	6
CHAPTER 2 - SOLUTION OF EQUATIONS		8
2.1	Introduction to Solution of Equations	8
2.2	Standard Gaussian Elimination	8
2.3	Gaussian Elimination as Matrix Decomposition	9
2.4	Banded Algorithms	10
2.5	Frontal Methods	12
2.6	Sparse Matrices	13
2.7	The Skyline Method	15
2.7.1	The Coefficient Matrix Storage Scheme	15
2.7.2	Columnwise 'Cholesky' Decomposition	18
2.8	Symbolic Development of Cholesky Decomposition for Substructure Analysis	20
2.9	Columnwise Algorithms for Substructure Displacement Analysis	24
2.9.1	Algorithm for $[L]^T$, $[D]$, and $\{R_i\}^*$	24
2.9.2	Algorithm for $[F]^T$	27
2.9.3	Algorithm for $[K]^*$, and $\{R_b\}^*$	29
2.9.4	Algorithm for Backsubstitution	31

2.9.5	The Equation Solving Package	33
2.10	Efficiency of the Skyline Method	33
CHAPTER 3 - ASSEMBLY AND COORDINATE TRANSFORMATION		35
3.1	Introduction to Assembly and Coordinate Transformation	35
3.2	Coordinate Transformation	35
3.3	Assembly	40
3.4	The Skyline of a Higher Level Unit	42
3.4.1	The Direct Approach	43
3.4.2	Skyline Modification	43
3.4.3	Predicting the Skyline	45
3.5	External Boundary Conditions	46
CHAPTER 4 - DEVELOPMENT OF PROGRAMS		47
4.1	Introduction to Program Development	47
4.2	SISAPF: A Program for Single-level Substructure Analysis of Plane Frames	48
4.2.1	Concepts and General Development of SISAPF	48
4.2.2	Logic Flow of SISAPF	52
4.3	MUSAPF: A Program for Multi-level Substructure Analysis of Plane Frames	56
4.3.1	Conceptual Development	56
4.3.2	General Description of MUSAPF	64
4.3.3	Logic Flow of MUSAPF	65
CHAPTER 5 - APPLICATION		69
5.1	Description of Structure	69

5.2 Partitioning Schemes 69
5.3 Size and Core Storage 78
5.4 CPU Time 80
5.5 Substructuring, and Nodal Numbering 80
5.6 Higher Level Unit Modified Skylines 81

CHAPTER 6 - SUMMARY AND CONCLUSIONS 84

REFERENCES 86
APPENDIX A - USER'S MANUAL FOR SISAPF 88
APPENDIX B - USER'S MANUAL FOR MUSAPF 104
APPENDIX C - PROGRAM DESCRIPTIONS AND LISTINGS 117
C.1 Program SISAPF 118
C.2 Program MUSAPF 127
C.3 The Data Managing Package 138
C.4 Data Storage and Retrieval 141
C.5 The Equation Solving Package 144
C.5.1 Description of Package 144
C.5.2 Description of Subroutines 144
C.5.3 Description of Arguments 145
C.6 The Formulation and Output Package 152
APPENDIX D - EXAMPLE INPUT 162

LIST OF TABLES

Table	Description	Page
1.1	Number of Numerical Operations Required to Obtain $[K]^*$, and $\{R_b\}^*$	5
4.1	Array NPG for Master System of the Example of Fig. 4.1	51
4.2	Array KSUB for Substructure Class 2 of the Example in Fig. 4.1	51
4.3	Arrays INDSUB, IRCSUB, and IBNSUB for IAU(5)	62
4.4	The Connectivity Array NPG for IAU(5)	62
4.5	Array LCSUB for IAU(5)	62
4.6	Backsubstitution Control	63
5.1	Control and Size Parameters for Examples 1 to 7	79
5.2	CPU Times for Execution	82

LIST OF FIGURES

FIGURE	DESCRIPTION	PAGE
1.1	Substructures	2
1.2	A Hierarchy of Substructures	2
2.1	Symmetric Banded Matrix	11
2.2	Progress of Decomposition	11
2.3	Advantage of Columnwise Storage	14
2.4	Sparseness Due to Substructure Nodal Numbering	14
2.5	Column Heights for an Element	16
2.6	Storage Scheme	17
2.7	Combined Algorithms for $[L]^T$, $[D]$, and $\{R_i\}^*$	26
2.8	Combined Algorithm for $[F]^T$	28
2.9	Combined Algorithm for $[K]^*$, and $\{R_b\}^*$	30
2.10	Algorithms for Backsubstitution	32
3.1	Reference Frames	36
3.2	Transformation Matrices	36
3.3	Orientation of Reference Frames	39
3.4	Transformed Submatrix $[Q]_{\ell m}$ (A plane frame problem)	39
3.5	The Assembly Scheme	41
3.6	Skyline 'Modification' Scheme	44
4.1	Details of Example for SISAPF	50
4.2	A Hierarchy of Substructures	59
4.3	Description of Hierarchial Loading	60-61
5.1	Details of Problem Structure	70
5.2	Substructure Scheme for Examples 2 and 7	71

5.3	Substructure Scheme for Example 3	72
5.4	Substructure Scheme for Example 4	73
5.5	Substructure Scheme for Example 5	74
5.6	Details of IBU(1) for Examples 2 and 7	75
5.7	Details of IBU(3) for Examples 2 and 7	76
5.8	Details of IBU(2) for Examples 2 and 7	77
5.9	Details of IAU(4) or the Master System for Examples 2 and 7	77

CHAPTER 1 - INTRODUCTION

1.1 Introductory Remarks

Matrix methods of structural analysis are based on representing the actual structure by discrete structural elements interconnected at a set of nodes. The matrices representing the elastic properties of the individual elements are assembled to model the total structure. This mathematical model is used to determine the forces and displacements at the joints. When these are known, the conditions within each element may be determined.

The substructure method of analysis is a natural development of this concept. A structure can be partitioned into a number of substructures, which are considered as complex structural elements. These substructures are assembled along the inter-boundary nodes (Fig. 1.1). Once the inter-boundary system is solved, the values of the unknowns associated with the internal degrees of freedom of the complex structural elements can be obtained.

The concept can be extended to model what might be called a hierarchy of substructures [5,4]. This means that a substructure may itself be an assemblage of small substructures, which may be assemblies of yet smaller units, until one reaches simple structural elements (Fig. 1.2).

There are two common methods of substructure analysis. The first method [11] obtains the boundary matrices through full release of the internal degrees of freedom. The second method [5,12] achieves the same result through decomposition, sometimes called partial release of the internal degrees of freedom. These two approaches are discussed

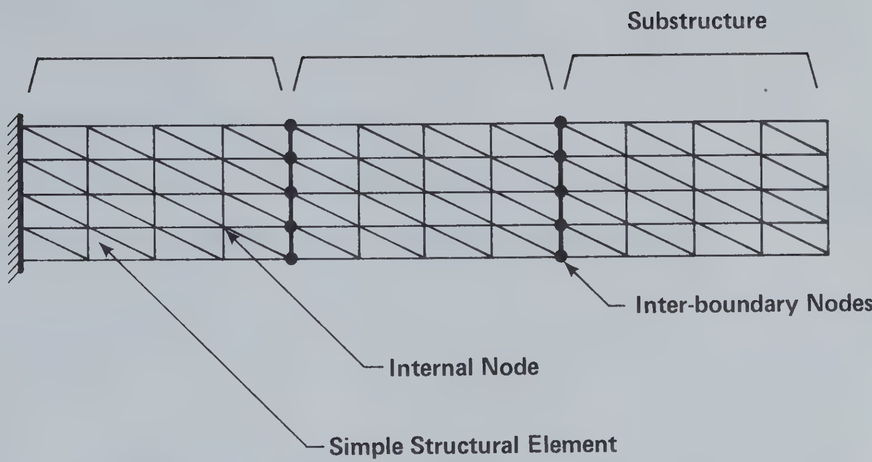


Figure 1.1 Substructures

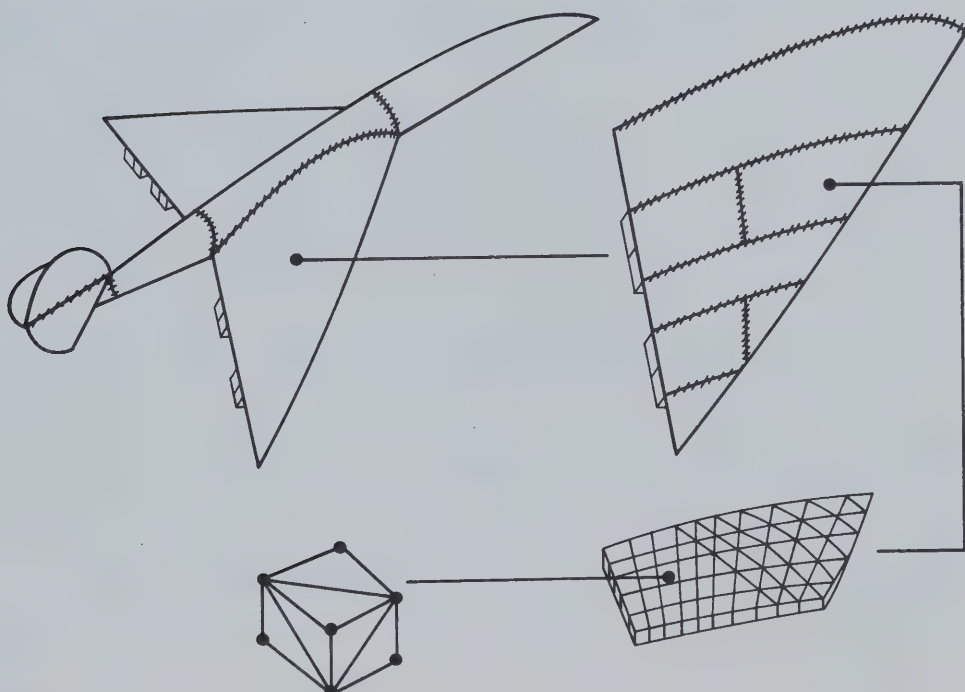


Figure 1.2 A Hierarchy of Substructures

in the following section.

1.2 Full Release vs Partial Release

Consider Eq. 1.1 which represents the load displacement relation of a substructure, without imposed boundary displacements,

$$[K] \ \{r\} = \{R\} \tag{1.1}$$

where $\{r\}$ is the vector of nodal displacements, $\{R\}$ is the vector of nodal forces, and [K] is the substructure stiffness matrix. Assume that the degrees of freedom can be partitioned into internal degrees of freedom, denoted by subscript i, and inter-boundary degrees of freedom, denoted by subscript b. Eq. 1.1 can be put in the form

$$\begin{bmatrix} K_{ii} & K_{ib} \\ K_{bi} & K_{bb} \end{bmatrix} \begin{Bmatrix} r_i \\ r_b \end{Bmatrix} = \begin{Bmatrix} R_i \\ R_b \end{Bmatrix} \tag{1.2}$$

From Eq. 1.2, $\{r_i\}$, and $\{r_b\}$ can be expressed as

$$\{r_i\} = [K_{ii}]^{-1} \ \{R_i\} - [K_{ib}] \ \{r_b\} \tag{1.3a}$$

$$\{r_b\} = [K^*]^{-1} \ \{R_b^*\} \tag{1.3b}$$

where,

$$[K^*] = [K_{bb}] - [K_{bi}]\,[K_{ii}]^{-1}\,[K_{ib}] \tag{1.3c}$$

and

$$\{R_b^*\} = \{R_b\} - [K_{bi}]\,[K_{ii}]^{-1}\,\{R_i\} \tag{1.3d}$$

If boundary conditions are imposed on $[K]^*$, the vector $\{r_b\}$ can be determined from Eq. 1.3b. Substituting this vector into Eq. 1.3a, the vector $\{r_i\}$ can also be obtained. This is called the 'full release' approach [11,12].

Another approach is to consider the component matrices of Eq. 1.2 as matrix elements, and carry out a simple Gaussian elimination on the first row, to get

$$\begin{bmatrix} K_{ii}^* & K_{ib}^* \\ 0 & K_{bb} - K_{bi}\,K_{ii}^{-1}\,K_{ib} \end{bmatrix} \begin{Bmatrix} r_i \\ r_b \end{Bmatrix} = \begin{Bmatrix} R_i^* \\ R_b - K_{bi}\,K_{ii}^{-1}\,R_i \end{Bmatrix} \tag{1.4}$$

The inter-boundary partitions of Eq. 1.4 are identical to those of Eqs. 1.3c, and 1.3d. Eq. 1.4 is formed by a decomposition of Eq. 1.2 up to the last row of the internal degrees of freedom [12,15]. This latter approach is the 'partial release' method, sometimes called the "Aitkin" decomposition [5].

Williams [15] has evaluated the number of numerical operations required to obtain the inter-boundary partitions using either approach. Table 1.1 simplifies his results for one case of loading. It can be seen that the partial release approach involves fewer numerical operations. Moreover, this table is very conservative in the case of partial

	Reciprocals	Multiplications	Additions (Subtractions)
Full Release	p	$(p^3+2p^2q+pq^2+4p^2+3pq-p-2)/2$	$(p^3+2p^2q+pq^2+p^2+pq-2p)/2$
Partial Release	p	$(p^3+3p^2q+3pq^2+6p^2+12pq-7p)/6$	$(p^3+3pq^2+3pq^2+3p^2+6pq-4p)/6$
Difference	o	$(2p^3+3p^2q+6p^2-3pq-3pq+4p-6)/6$	$(p^2-p)(2p+3p+2)/6$

Note: p: Number of internal degrees of freedom
q: Number of inter-boundary degrees of freedom

Table 1.1 Number of Numerical Operations Required to Obtain $[K]^*$ and $\{R_b^*\}$ (Matrices are assumed fully populated).

release since it assumes fully populated matrices. Because of the greater efficiency of the partial release technique the full release technique is seldom used and will not be discussed further herein.

1.3 Purpose and Scope

The objectives of this thesis are:

1) To develop an efficient equation solver to deal with the various stages of substructure analysis.
2) To develop a flexible single-level substructure analysis program, and a practical highly flexible multi-level substructuring scheme.
3) To explore the application of both single-level substructuring, and multi-level substructuring to planar framed structures.

1.4 Outline of Contents

Chapter 2 contains the theoretical background for, and the development of, an equation solving package suitable for substructure analysis. General techniques of equation solving are outlined. The 'skyline' method is developed for substructures, and the required algorithms are derived. The efficiency of the 'skyline' method is then compared to other methods in a qualitive way to indicate its advantages.

Considerations of assembly and coordinate transformation of the inter-boundary partitions are dealt with in Chapter 3. The logic flow and concepts of the single-level substructure analysis program SISAPF, followed by the conceptual development and the logic flow of the multi-level substructuring program MUSAPF, form the contents of Chapter 4.

In Chapter 5, an example problem is handled in several ways,

and the effects of the number, and the number of levels, of substructures on execution time are discussed. In Chapter 6 conclusions are drawn, and possible future developments are outlined.

CHAPTER 2 - SOLUTION OF EQUATIONS

2.1 Introduction to Solution of Equations

The equation solver is an essential part of any structural matrix analysis program. Several techniques have evolved since the advent of digital computers; all of them aimed at efficiency, since the CPU time consumed in equation solving usually constitutes 20% to 50% of the total time required to process a problem [9]. Reference 9 presents a review of these methods, and Reference 13 contains valuable information on the concepts involved in decomposition techniques. Some of the basic methods are reviewed in the following sections (Sect. 2.2 to 2.7). The skyline algorithms, developed by the author for implementation in the substructure analysis programs, form the subject material for the remainder of the Chapter.

2.2 Standard Gaussian Elimination

Consider the system of equations

$$[a] \quad \{x\} \quad = \quad \{b\} \tag{2.1}$$

in which [a] is a coefficient matrix, $\{x\}$ is the vector of unknowns, and $\{b\}$ is the (known), so-called, right hand side. The algorithm for elimination, with equation i as the pivotal equation, will yield

$$a_{jk} = a_{jk} - a_{ji} \cdot a_{ik}/a_{ii} \quad , \; k = i + 1, n \qquad , \; j = i + 1, n \tag{2.2a}$$

$$b_j = b_j - a_{ji} \cdot b_i/a_{ii} \ , \ j = i + 1, n \tag{2.2b}$$

The nth elimination will yield

$$x_n = b_n/a_{nn} \tag{2.3}$$

The unknowns x_{n-1}, x_{n-2}, x_1, are found by backsubstitution, using the equation

$$x_i = (b_i - \sum_{j=i+1}^{n} a_{ij} x_j)/a_{ii} \ , \ i = n-1, 1 \tag{2.4}$$

Introducing symmetry, which is a property of most structural matrices, $a_{ij} = a_{ji}$, and Eqs. 2.2a, and 2.2b become

$$a_{jk} = a_{jk} - a_{ij} \cdot a_{ik}/a_{ii} \ , \ k = j, n$$
$$, \ j = i + 1, n \tag{2.5a}$$

$$b_j = b_j - a_{ij} \, b_i/a_{ii} \ , \ j = i + 1, n \tag{2.5b}$$

Further, it should be noted that structural matrices for stable structures are always positive definite and well posed, hence pivoting is rarely justified [9] and will not be discussed.

2.3 Gaussian Elimination as Matrix Decomposition

Gaussian elimination can be interpreted as a decomposition of matrix [a] into upper and lower triangular matrices $[a]_\ell$, and $[a]_u$, such that

$$[a] = [a]_\ell \, [a]_u \tag{2.6}$$

The solution of Eq. 2.1 may be represented as

$$\{Y\} = [a]_\ell^{-1} \{b\} \tag{2.7a}$$

$$\{X\} = [a]_u^{-1} \{Y\} \tag{2.7b}$$

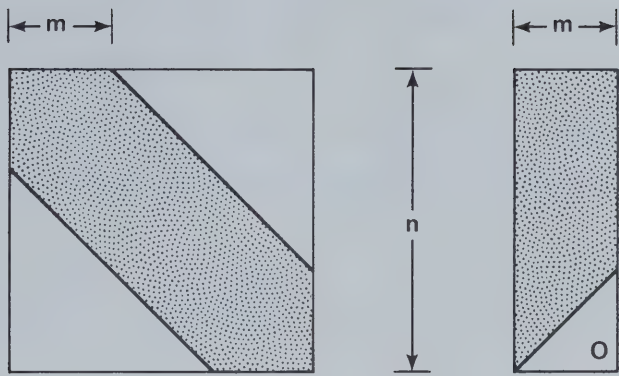
The coefficients of $[a]_u$ are those obtained from Eq. 2.2a. The diagonal components of $[a]_\ell$ are unity. The evaluation of $\{Y\}$ corresponds to the evaluation of $\{b\}$ by Eq. 2.2b or 2.5b, and the evaluation of $[a]_u^{-1} \{Y\}$ is the backsubstitution process described by Eq. 2.4. The Gaussian elimination process is thus a decomposition which proceeds by rows. It is possible to carry out the decomposition by columns, which is called Cholesky decomposition. For a symmetric matrix either decomposition can be put in the form

$$[a]_u^T [D] [a]_u = [a] \tag{2.8}$$

where the diagonal elements of $[a]_u$ in Eq. 2.8 are equal to unity, and $[D]$ is a diagonal matrix. This will be discussed in detail subsequently.

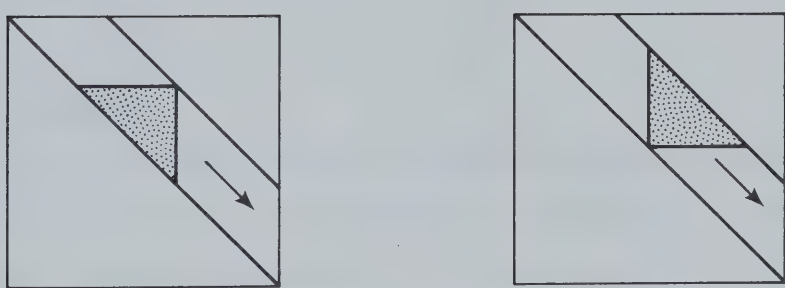
2.4 Banded Algorithms

It is possible in an unbranched problem to number the nodes, such that the non-zero components of the coefficient matrix are clustered about the main diagonal. This type of matrix is called 'banded', and



(a) (b)

Figure 2.1 Symmetric Banded Matrix



(a) Gauss (b) Cholesky

Figure 2.2 Progress of Decomposition [13]

the non-zero components of the matrix may be represented schematically as shown in Fig. 2.1a. The limits in Eq. 2.5a become $k = j, j + m - 1$, and $j = i + 1, i + m - 1$ in which m is called the 'half band width' and includes the main diagonal. Further, since the matrix is symmetric, only the half band need be stored in a rectangular matrix, see Fig. 2.1b. Eqs. 2.5, adjusted for this type of storage will become

$$a_{jk} = a_{jk} - a_{i,\ j-i+1} \cdot a_{i,\ k+j-i}/a_{i1} \ , \ k = 1, m + i - j \ , \ j = i + 1, i + m - 1 \quad (2.9a)$$

$$b_j = b_j - a_{i,\ j-i+1} \cdot b_i/a_{i1} \ , \ j = i + 1, i + m - 1 \quad (2.9b)$$

Fig. 2.2a shows the progress of a banded Gaussian elimination algorithm, where the shaded area is the space affected by elimination with equation i. Fig. 2.2b shows the progress of a banded Cholesky algorithm, and the shaded area is the space affected by decomposition of column i [7].

2.5 Frontal Methods

These methods are Gaussian elimination schemes in which a row is eliminated as soon as all its components are assembled. The order of assembly of the elements then governs the maximum band width arising in the solution algorithm. This method, despite its elegance, requires a complicated housekeeping system. The technique is effective in large systems, where efficient nodal numbering poses difficulties, and where the perepherial processing of the problem is more time consuming than solving the equations [9]. However, the maximum band width required in a wave front solution is the same as that required with an efficient

band solution.

2.6 Sparse Matrices

In sparse matrices, where the band is extremely irregular, pivotal rows will contain large percentages of zero components. These components are inactive (Fig. 2.3a), and can be skipped during numerical operations. However, in the case of a banded algorithm, they must be stored, unless the programmer introduces a separate addressing array. In a column, zero components below the first non-zero component may in general be expected to assume non-zero values during decomposition (Fig. 2.3a). A large saving in storage requirements can be achieved if only the active columns of a matrix are stored. The active column associated with degree of freedom j is defined, herein, as that portion of column j that begins at the first non-zero component and ends at the diagonal (Fig. 2.3b).

Columnwise decomposition is advantageous for the type of storage described above. In a substructured problem, where partial decomposition is required, the inter-boundary nodes should be grouped at one end of the nodal numbers, as indicated in Eq. 1.2, to avoid interchanging rows and columns before decomposition and after backsubstitution. Optimal nodal numbering, which gives a minimum band width is no longer feasible, and the resulting substructure stiffness matrix is sparse, as illustrated in Fig. 2.4.

For the reasons stated above, it was decided to use a columnwise decomposition scheme for the substructure analysis programs SISAPF, and MUSAPF. The method used is often referred to as the 'skyline method' [3,2], or a variable band width Cholesky method [13].

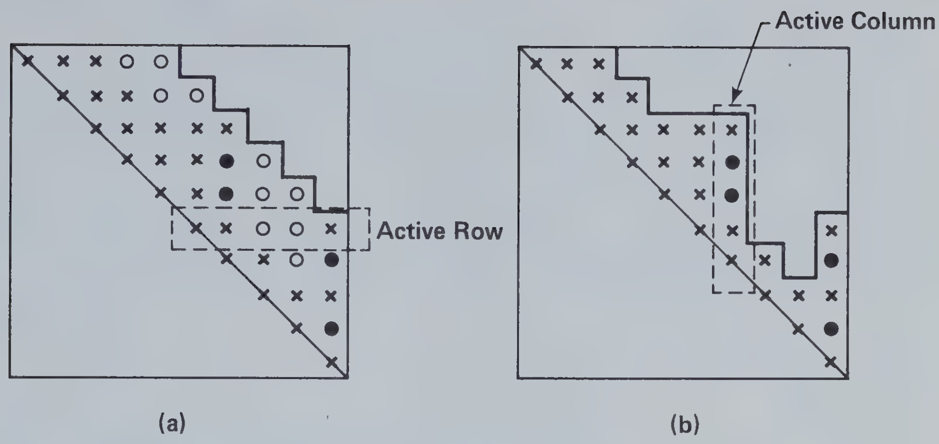


Figure 2.3 Advantage of Columnwise Storage

(× full components before and after decomposition
○ empty components before and after decomposition (inactive)
● components fill during decomposition)

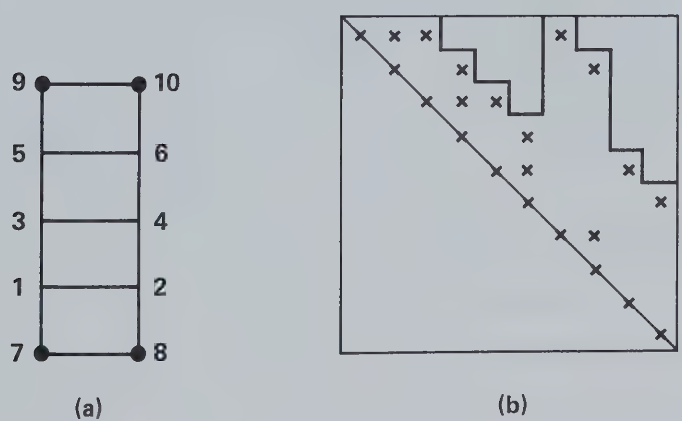


Figure 2.4 Sparseness Due to Substructure Nodal Numbering (Nodes 7, 8, 9, and 10 are the inter-boundary nodes.)

2.7 The Skyline Method

This method has been described by Wilson and Bathe [2], and by Felippa [3]. It has been used in some large scale finite element programs such as NONSAP [1], where it has been implemented for out of core solution of equations. It has two main features; the coefficient matrix storage scheme, and the columnwise decomposition algorithm. These are discussed below.

2.7.1 The Coefficient Matrix Storage Scheme

Since a structural stiffness matrix is symmetrical, only the upper triangle, including the main diagonal need be stored, and then only the active columns need be stored. The active column height for any degree of freedom is the difference between this degree of freedom and the lowest degree of freedom that appears in any element associated with it, see Fig. 2.5. It is tacitly assumed that all degrees of freedom associated with any one element are inter-active. By looping over all elements, determining the difference described above, and updating column heights as may be necessary, the skyline of the structure stiffness matrix is established. The skyline is defined as a bound on the active column heights (Fig. 2.6a).

The active columns are stored in a one-dimensional array, see Fig. 2.6b. Access to any component is ensured if the addresses of the diagonal components are known. These are obtained by adding up the column heights, and storing the addresses of the diagonal components in a separate array, MAXA (Fig. 2.6c). For example, access to component a_{ij}, where i is the row number, and j is the column number is

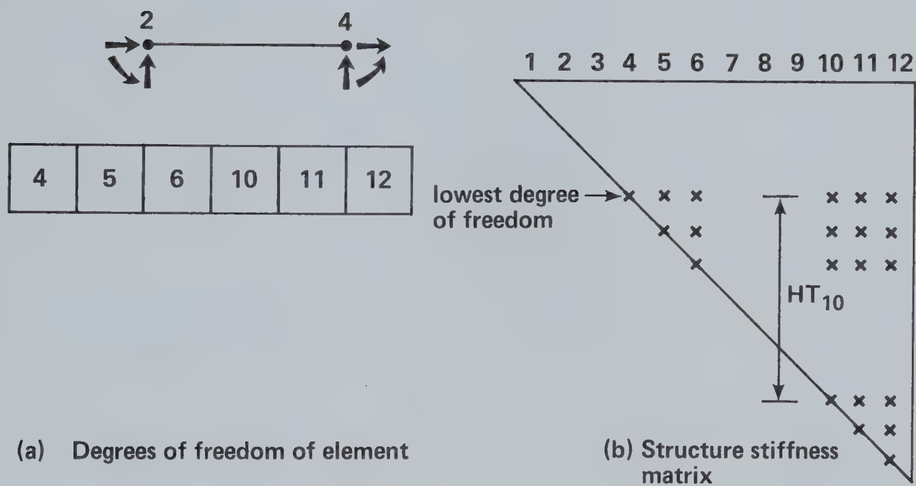


Figure 2.5 Column Heights for An Element ($HT_{10} \equiv$ Column height of degree of freedom 10 due to connectivity of element shown in (a))

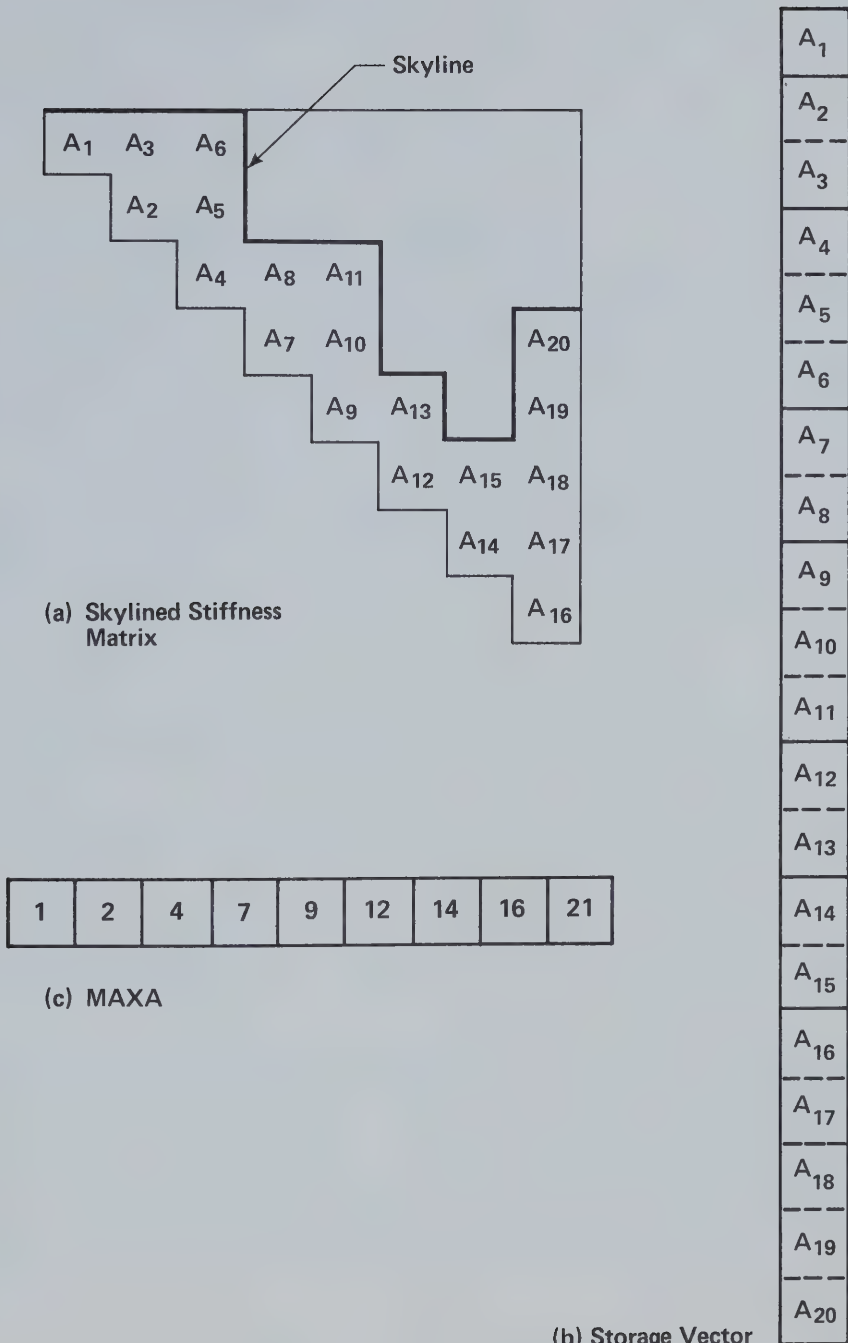


Figure 2.6 Storage Scheme

$$a_{ij} = A_m \, , \; m = MAXA_j + j - i \tag{2.10}$$

and A is the one dimensional storage array.

2.7.2 Columnwise 'Cholesky' Decomposition

This summary of the columnwise decomposition has been largely drawn from Reference 2. In the following the stiffness matrix will be denoted by [K], the load vector by $\{R\}$, and the displacement vector by $\{U\}$. The matrix corresponding to $[a]_u^T$ of Eq. 2.8 will be denoted by [L]. The stiffness matrix is symmetric, and if enough restraints are imposed on the system, it is positive definite. Hence, Eq. 2.1 can be put in the form

$$[L]\,[D]\,[L]^T \;\{U\} \;=\; \{R\} \tag{2.11}$$

or

$$[L]\,[G]\;\{U\} \;=\; \{R\} \tag{2.12a}$$

where

$$[G] = [D]\,[L]^T \tag{2.12b}$$

Matrices [L], [G], and [D] are not explicitly obtained. Instead, the coefficients of [K] are modified. Eqs. 2.12a, and 2.12b yield the following algorithm, where the subscripts apply to a square matrix, and m_j is the row subscript of the first non-zero element appearing in column j:

For all j, j = 1, n

$$K_{ij} = K_{ij} - \sum_{s=m_j}^{i-1} K_{si}\, K_{sj} \ , \ i = m_j + 1, j - 1 \qquad (2.13a)$$

$$K_{ij} = K_{ij}/K_{ii} \ , \ i = m_j, j - 1 \qquad (2.13b)$$

$$K_{jj} = K_{jj} - \sum_{s=m_j}^{j-1} K_{ss}\, K_{sj} \qquad (2.13c)$$

Matrices [G], $[L]^T$ and [D] are formed by Eqs. 2.13a, 2.13b, and 2.13c respectively. The solution is then obtained as

$$\{U\} = [L^T]^{-1}\, [D]^{-1}\, \{V\} \qquad (2.14a)$$

where

$$\{V\} = [L]^{-1}\, \{R\} \qquad (2.14b)$$

Eq. 2.14b yields the forward substitution algorithm described by

$$V_j = R_j - \sum_{i=m_j}^{j-1} L_{ij}\, V_i \ , \ j = 2, n \qquad (2.15a)$$

In coding this algorithm, the vector {V} overwrites {R} resulting in

$$R_j = R_j - \sum_{i=m_j}^{j-1} K_{ji}\, R_i \ , \ j = 2, n \qquad (2.15b)$$

Eq. 2.14a describes the backsubstitution process and can be put in the form

$$V_j = V_j/D_{jj} = V_j/K_{jj} \ , \ j = 1, n \qquad (2.16)$$

For j = n the new V_n becomes U_n of Eq. 2.14a.

For all i, i = n, 2

$$V_j^{(i-1)} = V_j^{(i)} - K_{ji} V_i^{(i)} , \quad j = m_i, i - 1 \tag{2.17}$$

where the superscript indicates that variable is computed in backsubstitution step number (n - superscript), and the final values of V_j of Eq. 2.17 are the vector {U} of Eq. 2.14a.

2.8 Symbolic Development of Cholesky Decomposition for Substructure Analysis

In order to determine the effective inter-boundary stiffness matrix $[K]^*$ by the partial release method it is required to reduce Eq. 1.2 to the form

$$\begin{bmatrix} G^* & \overline{K} \\ 0 & K^* \end{bmatrix} \begin{Bmatrix} U_i \\ U_b \end{Bmatrix} = \begin{Bmatrix} R_i^{**} \\ R_b^{*} \end{Bmatrix} \tag{2.18}$$

where $[G]^*$ is an upper triangular matrix, and i and b denote the internal and inter-boundary degrees of freedom respectively. It will be shown that $[K]^*$ of Eq. 2.18 is identical to $[K]^*$ of Eq. 1.3c. Since [K] is symmetric it can be decomposed to the form $[L'] [D'] [L']^T$, in which $[D']$ is composed of 4 submatrices, where the two off-diagonal submatrices are null matrices, and one of the partitions must have enough restraints

to stabilize the system. To demonstrate this, assume that the matrix product $[L'] [D'] [L']^T$ can be written in the form [12]

$$\begin{bmatrix} L & 0 \\ F & I \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & K^* \end{bmatrix} \begin{bmatrix} L^T & F^T \\ 0 & I \end{bmatrix} \begin{Bmatrix} U_i \\ U_b \end{Bmatrix} = \begin{Bmatrix} R_i \\ R_b \end{Bmatrix} \tag{2.19}$$

Then the product of the matrices in Eq. 2.19 may be written as

$$\begin{bmatrix} L\,D\,L^T & L\,D\,F^T \\ F\,D\,L^T & K^* + F\,D\,F^T \end{bmatrix} \begin{Bmatrix} U_i \\ U_b \end{Bmatrix} = \begin{Bmatrix} R_i \\ R_b \end{Bmatrix} \tag{2.20}$$

If the matrix of Eq. 2.20 represents the stiffness matrix, the submatrices must identify with those of Eq. 1.2. This establishes the requirements

$$[K_{ii}] = [L]\,[D]\,[L]^T \tag{2.21a}$$

$$[K_{ib}] = [L]\,[D]\,[F]^T \tag{2.21b}$$

$$[K_{bb}] = [K]^* + [F]\,[D]\,[F]^T \tag{2.21c}$$

Multiplying Eq. 2.20 by $\begin{bmatrix} D^{-1}\,L^{-1} & 0 \\ -F\,L^{-1} & I \end{bmatrix}$, we get

$$\begin{bmatrix} L^T & F^T \\ 0 & K^* \end{bmatrix} \begin{Bmatrix} U_i \\ U_b \end{Bmatrix} = \begin{Bmatrix} D^{-1}\,L^{-1}\,R_i \\ R_b - F\,L^{-1}\,R_i \end{Bmatrix} \tag{2.22}$$

which is of the required form. Eq. 2.21a indicates that $[L]^T$ and $[D]$ may be determined from a standard decomposition of $[K_{ii}]$. Once this has been accomplished $[F]^T$ may be determined from Eq. 2.21b as

$$[F]^T = [D]^{-1} \, [L]^{-1} \, [K_{ib}] \tag{2.23a}$$

and from Eq. 2.21c

$$[K]^* = [K_{bb}] - [F] \, [D] \, [F]^T \tag{2.23b}$$

Using the relation of Eq. 2.23a to substitute for $[F]$ and $[F]^T$ in Eq. 2.23b we get

$$[K]^* = [K_{bb}] - [K_{ib}]^T \, [L]^{-1^T} \, [D]^{-1} \, [D] \, [D]^{-1} \, [L]^{-1} \, [K_{ib}]$$

or

$$[K]^* = [K_{bb}] - [K_{bi}] \, [K_{ii}]^{-1} \, [K_{ib}] \tag{2.23c}$$

which proves that matrix $[K]^*$ of Eq. 2.23b is identical to $[K]^*$ of Eq. 1.3c.

To demonstrate that the right hand side of Eq. 2.22 is equivalent to the right hand side of Eq. 1.3d and 1.4, call the inter-boundary force partition of Eq. 2.22 $\{R_b\}^*$, then

$$\{R_b\}^* = \{R_b\} - [F] \, [L]^{-1} \, \{R_i\} \tag{2.23d}$$

now substitute for $[F]$ from Eq. 2.23a into Eq. 2.23d

$$\{R_b\}^* = \{R_b\} - [K_{ib}]^T [L]^{-1T} [D]^{-1} [L]^{-1} \{R_i\}$$

or

$$\{R_b\}^* = \{R_b\} - [K_{bi}] [K_{ii}]^{-1} \{R_i\} \tag{2.23e}$$

Vector $\{R_b\}^*$ of Eq. 2.23e is identical to $\{R_b\}^*$ of Eq. 1.3d.

For obvious practical considerations, submatrices $[L]^T$, $[F]^T$, $[D]$, $[K]^*$, and $\{R_b\}^*$ will be obtained by modifying the coefficients of $[K]$, and $\{R\}$. The step $[D]^{-1} [L]^{-1} \{R_i\}$ in Eq. 2.22 will be restricted to $[L]^{-1} \{R_i\}$, since this is all that is needed until backsubstitution begins.

The relations required for backsubstitution can be obtained by carrying out the multiplications involved in Eq. 2.22. Thus, from the first row of Eq. 2.22

$$[D]^{-1} \{R_i\}^* = [L]^T \{U_i\} + [F]^T \{U_b\} \tag{2.24a}$$

where

$$\{R_i\}^* = [L]^{-1} \{R_i\} \tag{2.24b}$$

Assuming that vector $\{U_b\}$ will be obtained from a solution of the inter-boundary system, vector $\{U_i\}$, which represents the displacements of the internal nodes, can be obtained from Eq. 2.24a as

$$\{V_i\} = [D]^{-1} \{R_i\}^* - [F]^T \{U_b\} \tag{2.25}$$

and

$$\{U_i\} = [L]^{T\,-1} \{V_i\} \tag{2.26}$$

If the problem is not substructured, $[K_{ii}]$, $\{U_i\}$, and $\{R_i\}$

default to [K], {U} , and {R} respectively, and Eqs. 2.20, and 2.26 default to Eqs. 2.11 and 2.14a

2.9 Columnwise Algorithms for Substructure Displacement Analysis

In this section the algorithms necessary to carry out the partial release procedure described in Section 2.8 are derived. A number of algorithms can be derived for Eqs. 2.21, 2.25, and 2.26. If a skyline is imposed on the stiffness matrix, the resulting algorithms form the basis of a highly efficient equation solving package, suitable for substructure analysis. The derivation of the various algorithms, along with their final form after imposing the skyline on the stiffness matrix, follows.

2.9.1 Algorithm for $[L]^T$, [D], and $\{R_i\}^*$

Consider a 6x6 substructure stiffness matrix, where degrees of freedom 5, and 6 form the inter-boundary partition. Eq. 2.21a can be put in the form

$$\begin{bmatrix} 1 & & & \\ L_{21} & 1 & 0 & \\ L_{31} & L_{32} & 1 & \\ L_{41} & L_{42} & L_{43} & 1 \end{bmatrix} \begin{bmatrix} d_{11}, & d_{11}\,L_{21}, & d_{11}\,L_{31}, & d_{11}\,L_{41} \\ & d_{22} & d_{22}\,L_{32}, & d_{22}\,L_{42} \\ & 0 & d_{33} & d_{33}\,L_{43} \\ & & & d_{44} \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ & K_{22} & K_{23} & K_{24} \\ & & K_{33} & K_{34} \\ & & & K_{44} \end{bmatrix} \tag{2.27}$$

Expanding the fourth column will yield the set of equations

$$K_{14} = d_{11}\,L_{41} \tag{2.28a}$$

$$K_{24} = d_{11} L_{41} L_{21} + d_{22} L_{42} \tag{2.28b}$$

$$K_{34} = d_{11} L_{41} L_{31} + d_{22} L_{42} L_{32} + d_{33} L_{43} \tag{2.28c}$$

$$K_{34} = d_{11} L_{41} L_{31} + d_{22} L_{42} L_{42} + d_{33} L_{43} L_{43} + d_{44} \tag{2.28d}$$

Assuming that L_{ij} is known for i = 1,4, and j = 1,2, Eq. 2.28c can be solved for L_{43} as follows

$$G_{34} = d_{33} L_{43} = K_{34} - \sum_{S=1}^{2} G_{S4} L_{3S} \tag{2.29a}$$

$$L_{43} = G_{34}/d_{33} \tag{2.29b}$$

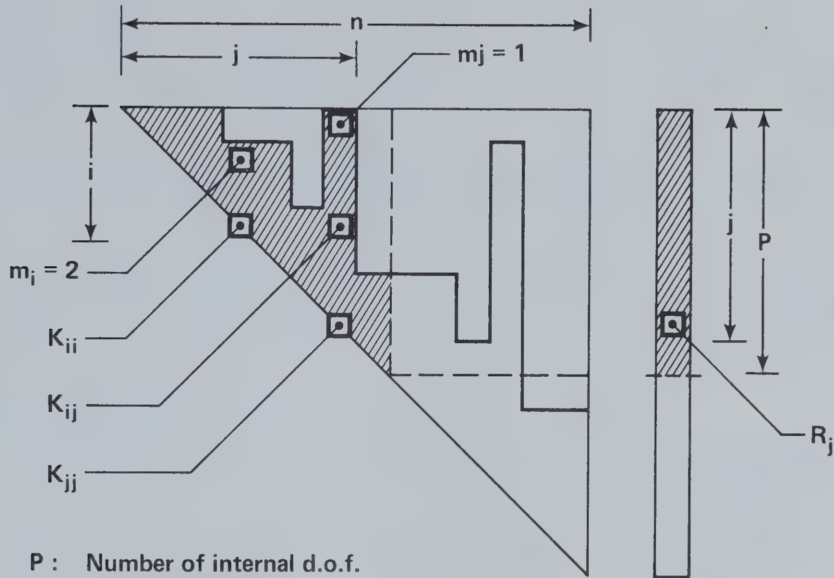
and Eq. 2.28d can be solved for d_{44}

$$d_{44} = K_{44} - \sum_{S=1}^{3} G_{S4} L_{4S} \tag{2.29c}$$

Generalizing Eqs. 2.29 yields the algorithm to determine $[L]^T$ and [D].

Eqs. 2.24b can be put in the form $\{R_i\} = [L] \{R_i\}^*$, which is expanded for the above substructure as follows

$$\begin{bmatrix} 1 & & & \\ L_{21} & 1 & & \\ L_{31} & L_{32} & 1 & \\ L_{41} & L_{42} & L_{43} & 1 \end{bmatrix} \begin{Bmatrix} R_1^* \\ R_2^* \\ R_3^* \\ R_4^* \end{Bmatrix} = \begin{Bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{Bmatrix} \tag{2.30}$$



P : Number of internal d.o.f.

n : Total number of d.o.f.

mm: Maximum of m_i, m_j

For j = 2, p

$$K_{ij} = K_{ij} - \sum_{s=mm}^{i-1} K_{sj}K_{si} \quad , i = m_j + 1, j-1$$

$$K_{ij} = K_{ij}/K_{ii} \quad , i = m_j, j-1$$

$$K_{jj} = K_{jj} - \sum_{s=mj}^{j-1} K_{ss}K_{sj}^2$$

$$R_j = R_j - \sum_{s=mj}^{j-1} K_{sj}R_s$$

Figure 2.7 Combined Algorithm for $[L]^T$, $[D]$, and $\{R_i^*\}$

Expanding the fourth row of Eq. 2.30 will yield

$$R_4 = R_1^* L_{41} + R_2^* L_{42} + R_3^* L_{43} + R_4^* \qquad (2.31)$$

which can be solved for R_4^*, if R_i^* for i = 1,3 are known. Thus,

$$R_4^* = R_4 - \sum_{S=1}^{3} R_S^* L_{4S} \qquad (2.32)$$

which when generalized, yields an algorithm for $\{R_i\}^*$.

Notice that when Eqs. 2.29 and 2.32 are implemented on the computer L_{4S} will overwrite K_{S4}, and d_{44} will overwrite K_{44}. These equations can be combined into the algorithm shown in Fig. 2.7.

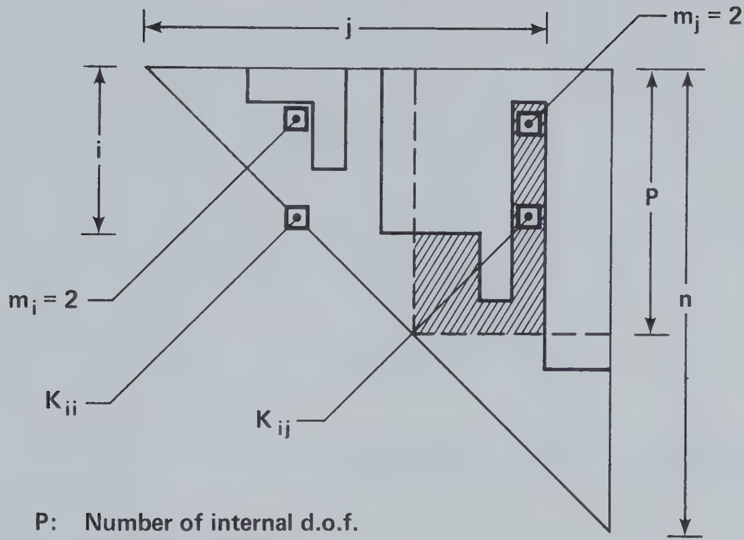
2.9.2 Algorithm for $[F]^T$

With $[L]^T$ and [D] determined, as in Sect. 2.9.1, Eq. 2.21b can be expanded as follows,

$$\begin{bmatrix} d_{11} & & & \\ d_{11} L_{21}, & d_{22} & & 0 \\ d_{11} L_{31}, & d_{22} L_{32}, & d_{33} & \\ d_{11} L_{41}, & d_{22} L_{42}, & d_{33} L_{43}, & d_{44} \end{bmatrix} \begin{bmatrix} \bar{K}_{15} & \bar{K}_{16} \\ \bar{K}_{25} & \bar{K}_{26} \\ \bar{K}_{35} & \bar{K}_{36} \\ \bar{K}_{45} & \bar{K}_{46} \end{bmatrix} = \begin{bmatrix} K_{15} & K_{16} \\ K_{25} & K_{26} \\ K_{35} & K_{36} \\ K_{45} & K_{46} \end{bmatrix} \qquad (2.33)$$

where the $\bar{K}_{\ell m}$ represent the elements of $[F]^T$. This notation is used because the elements of $[F]^T$ overwrite the $[K_{ib}]$ partition of the stiffness matrix. The expansion of the second column will yield

$$K_{16} = \bar{K}_{16} \, d_{11} \qquad (2.34a)$$



P: Number of internal d.o.f.
n: Total number of d.o.f.
mm: Maximum of m_i, m_j

For all j, $j = p + 1, n$

$$K_{ij} = K_{ij} - \sum_{s=mm}^{i-1} K_{sj} K_{si} \qquad , i = mj + 1, p$$

$$K_{ij} = K_{ij} / K_{ii} \qquad , i = mj, p$$

Figure 2.8 Algorithm for $[F]^T$

$$K_{26} = \overline{K}_{16}\, d_{11}\, L_{21} + \overline{K}_{26}\, d_{22} \qquad (2.34b)$$

$$K_{36} = \overline{K}_{16}\, d_{11}\, L_{31} + \overline{K}_{26}\, d_{22}\, L_{32} + \overline{K}_{36}\, d_{33} \qquad (2.34c)$$

$$K_{46} = \overline{K}_{16}\, d_{11}\, L_{41} + \overline{K}_{26}\, d_{22}\, L_{42} + \overline{K}_{36}\, d_{33}\, L_{43} + \overline{K}_{46}\, d_{44} \qquad (2.34d)$$

Eq. 2.34d can be solved for $\overline{K}_{46}$ if $\overline{K}_{i6}$, i = 1,3 are known.

$$\overline{K}_{46} = (K_{46} - \sum_{S=1}^{3} \overline{K}_{S6}\, K_{SS}\, K_{S4})/K_{44} \qquad (2.35)$$

Eq. 2.35b, when generalized, can be expressed by the algorithm shown in Fig. 2.8.

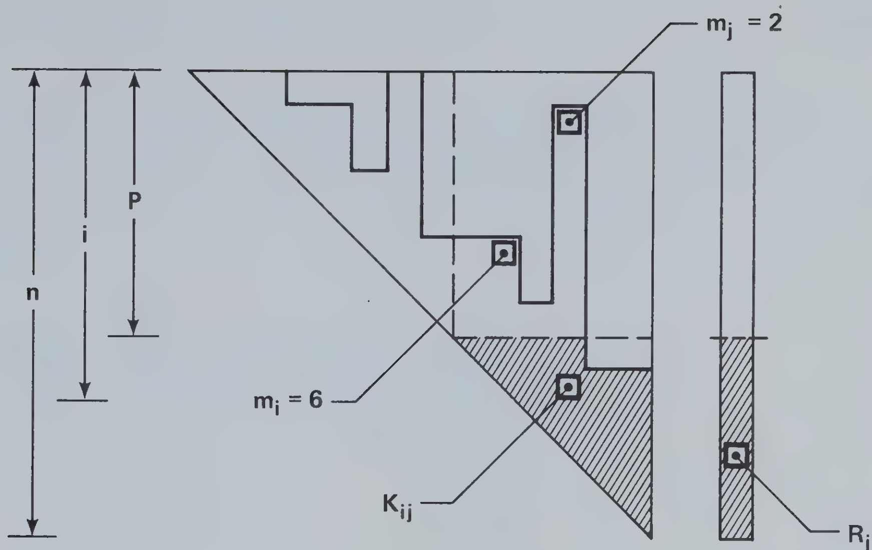
2.9.3 Algorithm for $[K]^*$, and $\{R_b\}^*$

With $[F]^T$ known and represented by $[\overline{K}]$, the components of $[K]^*$ may be determined by expanding Eq. 2.21c

$$\begin{bmatrix} K^*_{55} & K^*_{56} \\ K^*_{65} & K^*_{66} \end{bmatrix} = \begin{bmatrix} K_{55} & K_{56} \\ K_{65} & K_{66} \end{bmatrix} - \begin{bmatrix} d_{11}\,\overline{K}_{15}, & d_{22}\,\overline{K}_{25}, & d_{33}\,\overline{K}_{35}, & d_{44}\,\overline{K}_{45} \\ d_{11}\,\overline{K}_{16}, & d_{22}\,\overline{K}_{26}, & d_{33}\,\overline{K}_{36}, & d_{44}\,\overline{K}_{46} \end{bmatrix} \begin{bmatrix} \overline{K}_{15} & \overline{K}_{16} \\ \overline{K}_{25} & \overline{K}_{26} \\ \overline{K}_{35} & \overline{K}_{36} \\ \overline{K}_{45} & \overline{K}_{46} \end{bmatrix} \qquad (2.36)$$

Component K^*_{56} can be expressed as

$$K^*_{56} = K_{56} - [d_{11}\,\overline{K}_{15}\,\overline{K}_{16} + d_{22}\,\overline{K}_{25}\,\overline{K}_{26} + d_{33}\,\overline{K}_{35}\,\overline{K}_{36} + d_{44}\,\overline{K}_{45}\,\overline{K}_{46}] \qquad (2.37a)$$



P: Number of internal d.o.f.
n: Total number of d.o.f.
mm: Maximum of m_i, m_j

For all j, $j = p + 1, n$

$$K_{ij} = K_{ij} - \sum_{s = mm}^{P} K_{ss} K_{si} K_{sj} \quad , i = (\text{max. of } (p + 1), mj), j$$

$$R_j = R_j - \sum_{s = mj}^{P} K_{sj} R_s$$

Figure 2.9 The Combined Algorithm for $[K]^*$, and $\left\{R_b^*\right\}$

or

$$K^*_{56} = K_{56} - \sum_{S=1}^{4} K_{SS}\,\overline{K}_{S5}\,\overline{K}_{S6} \qquad (2.37b)$$

With $\{R_i\}^*$ determined as in Section 2.9.1, and $[F]^T$ determined in Section 2.9.2, $\{R_b\}^*$ can be determined by expanding Eq. 2.23d as follows

$$\begin{Bmatrix} R^*_5 \\ R^*_6 \end{Bmatrix} = \begin{Bmatrix} R_5 \\ R_6 \end{Bmatrix} - \begin{bmatrix} \overline{K}_{15} & \overline{K}_{25} & \overline{K}_{35} & \overline{K}_{45} \\ \overline{K}_{16} & \overline{K}_{26} & \overline{K}_{36} & \overline{K}_{46} \end{bmatrix} \begin{Bmatrix} R^*_1 \\ R^*_2 \\ R^*_3 \\ R^*_4 \end{Bmatrix} \qquad (2.38)$$

Component R^*_6 can be expressed as

$$R^*_6 = R_6 - [\overline{K}_{16}\,R^*_1 + \overline{K}_{26}\,R^*_2 + \overline{K}_{36}\,R^*_3 + \overline{K}_{46}\,R^*_4] \qquad (2.39a)$$

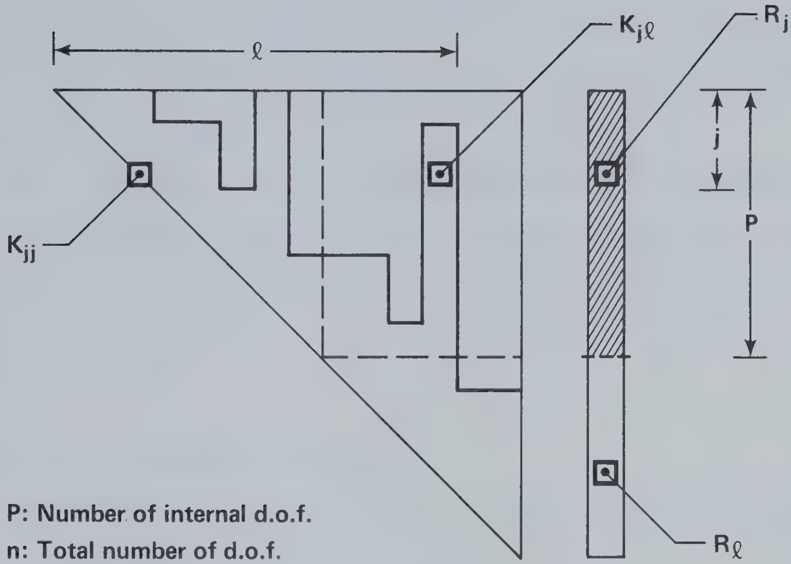
or

$$R^*_6 = R_6 - \sum_{S=1}^{4} K_{S6}\,R^*_S \qquad (2.39b)$$

A combined algorithm representing Eqs. 2.37b, and 2.39b is shown in Fig. 2.9. The inter-boundary stiffness matrix $[K]^*$ of Eq. 2.22 and the associated force vector $\{R_b\}^*$ of Eq. 2.23e, have therefore been determined.

2.9.4 Algorithms for Backsubstitution

Submatrices $[K]^*$ and $\{R_b\}^*$ are the inter-boundary partitions. The assembly of these partitions from a number of substructures forms a higher level substructure unit in what has been called a hierarchy of



$R_j = R_j / K_{jj}$, $j = 1, p$

For all ℓ, $\ell = p + 1, n$

$R_j = R_j - K_{j\ell} R_\ell$, $j = m_j , p$

For all j, $j = p, 2$

$R_s^{(j-1)} = R^{(j)}_s - K_{sj} R_j$, $s = mj, j - 1$

$R_{j-1} = R_{j-1}^{(j-1)}$

Figure 2.10 Algorithms for Backsubstitution

substructures. When the summit of the heirachy, referred to later as the 'master system', has been formed, backsubstitution to obtain the final inter-boundary nodal displacements, and subsequently the internal nodal displacements, begins.

Eqs. 2.25, and 2.26 have been expanded to the algorithm shown in Fig. 2.10, where the upper part of the algorithm evaluates $\{V_i\}$ of Eq. 2.25, and the lower part of the algorithm is the backsubstitution of Eq. 2.26.

2.9.5 The Equation Solving Package

The algorithms developed above have been coded into a set of subroutines for solving substructured problems. The subroutines, along with their listing are described in Appendix C, and their use within the programs is described in Chapter 4.

2.10 Efficiency of the Skyline Method

The efficiency of this method relative to other methods is not readily determined in terms of the number of numerical operations since it is dependent on the skyline of the sparse matrix, and on the partitions adopted. However, several aspects of the method indicate that a substantial reduction of CPU time can be achieved. The saving in storage due to exclusion of zero wedges is obvious. The elimination of these wedges further eliminates the need to check for zero elements in the outer loops of the coding. Moreover, the method, as coded, automatically adjusts for the smaller vector when evaluating the scalar product of two vectors, whereas in banded algorithms, the vector that appears in the

outer loop governs the length of vector products for the next m operations.

The fact that the elements required in evaluating the vector products are accessed in sequential storage locations indicates that the technique will operate efficiently in virtual memory machines, and the singly subscripted arrays may also be advantageous in some computers.

CHAPTER 3 - ASSEMBLY AND COORDINATE TRANSFORMATION

3.1 Introduction to Assembly and Coordinate Transformation

As shown in Fig. 3.1, a local reference frame is associated with each substructure. The position of a substructure in a higher level assembled system is defined by the inter-boundary nodal connectivity, and by the orientation of the local substructure reference frame relative to the higher level reference frame. The assembly of substructure inter-boundary partitions into a higher level system must take care of both connectivity and coordinate transformation.

Two assumptions were made to simplify the assembly process. The first, already mentioned in Chapter 2, is that the inter-boundary nodes of a substructure are grouped at the end of its local nodal numbers. The second, is that there will be a fixed number of degrees of freedom per node, and that the local ordering of these degrees of freedom will be constant throughout the problem. The second assumption implies that a stiffness matrix can be partitioned into a set of qxq submatrices, where q is the number of degrees of freedom per node. Let these submatrices form the elements of a matrix [Q] equivalent to the stiffness matrix. By definition, component $[Q]_{\ell m}$ is the qxq set of forces at node ℓ due to a set of unit displacements applied separately at node m, when all other sets of displacements are zero.

3.2 Coordinate Transformation

Let vectors $\{U_b\}$ and $\{R^*_b\}$ of Eq. 2.18 be the displacement and force interboundary partitions of a substructure in its local reference frame, and let vectors $\{U_h\}$ and $\{R_h\}$ be the corresponding

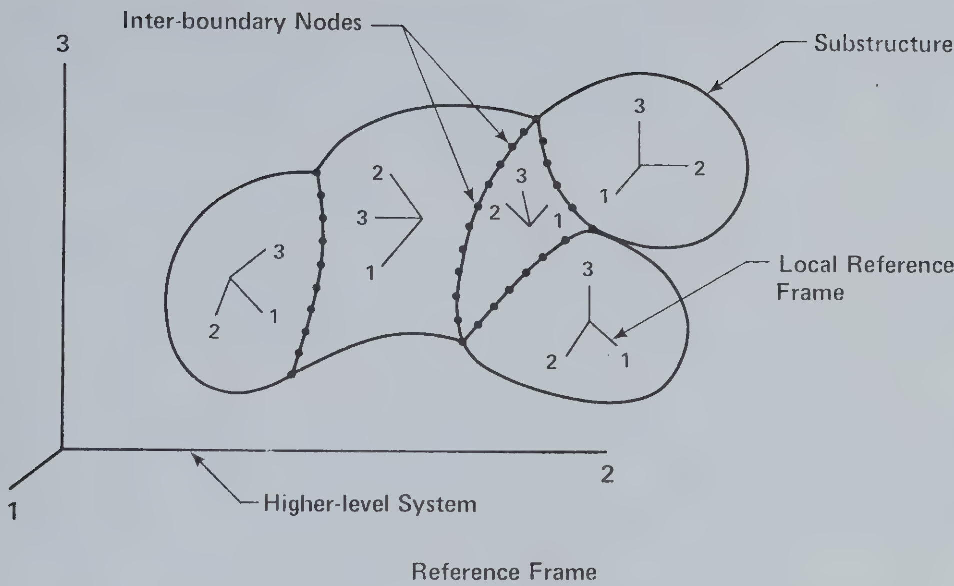


Figure 3.1 Reference Frames

$$
\begin{bmatrix}
[\bar{T}_1] & & & & \\
& [\bar{T}_1] & & O & \\
& & [\bar{T}_1] & & \\
& O & & [\bar{T}_1] & \\
& & & & [\bar{T}_1]
\end{bmatrix}
\qquad
\begin{bmatrix}
[\bar{T}_2] & & & & \\
& [\bar{T}_2] & & O & \\
& & [\bar{T}_2] & & \\
& O & & [\bar{T}_2] & \\
& & & & [\bar{T}_2]
\end{bmatrix}
$$

$$
[T_1] \qquad\qquad\qquad [T_2]
$$

Figure 3.2 Transformation Matrices

vectors in a higher level system reference frame. The orthogonal transformation between the two systems takes the form

$$\{U_b\} = [T_1] \ \{U_h\} \tag{3.1a}$$

$$\{R_h\} = [T_2] \ \{R^*_b\} \tag{3.1b}$$

where $[T_1]$ and $[T_2]$ are each composed of n^2 submatrices of order (qxq) and n is the number of inter-boundary nodes. These submatrices are null everywhere except those on the main diagonals of $[T_1]$ and $[T_2]$, denoted respectively as $[\overline{T}_1]$ and $[\overline{T}_2]$ as illustrated in Fig. 3.2.

Let u, v, and r be the local displacement components at a node in a plane frame problem, where u is the displacement component in the local X-direction, v is the displacement component in the local Y-direction, and r is an anti-clockwise rotation; and u_h, v_h, and r_h be the corresponding components at the same node in a higher level system reference frame. Let R_u, R_v, and R_r be the force components at a node in the local reference frame, corresponding to degrees of freedom u, v, and r respectively, and let F_u, F_v, and F_r be the force components at the same node in a higher level system reference frame. The orthogonal transformations between the two sets of free vectors takes the form

$$\{U_b\} = \begin{Bmatrix} u \\ v \\ r \end{Bmatrix} = \begin{bmatrix} C & S & 0 \\ -S & C & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} u_h \\ v_h \\ r_h \end{Bmatrix} = [\overline{T}_1] \ \{U_h\} \tag{3.2a}$$

and

$$\{R_h\} = \begin{Bmatrix} F_u \\ F_v \\ F_r \end{Bmatrix} = \begin{bmatrix} C & -S & 0 \\ S & C & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} R_u \\ R_v \\ R_h \end{Bmatrix} = [\bar{T}_2] \ \{R^*_b\} \qquad (3.2b)$$

where C is the cosine of angle θ, and S is the sine of θ, as illustrated in Fig. 3.3. Consequently, from Eqs. 3.1 and 3.2,

$$[\bar{T}_1] = \begin{bmatrix} C & S & 0 \\ -S & C & 0 \\ 0 & 0 & 1 \end{bmatrix} = [\bar{T}_2]^T = [\bar{T}] \qquad (3.3)$$

and the assembled transformation matrices for the inter-boundary nodes become

$$[T_2]^T = [T_1] = [T] \qquad (3.4)$$

The second row of Eq. 2.18 can be written in the form

$$[K^*] \ [T] \ \{U_h\} = \{R^*_b\} \qquad (3.5)$$

Premultiplying Eq. 3.5 by $[T]^T$ we get

$$[T]^T \ [K^*] \ [T] \ \{U_h\} = [T]^T \ \{R^*_b\} = \{F\} \qquad (3.6)$$

where $\{F\}$ is the vector of effective forces in the higher level reference frame. Since $[K^*]$ is symmetric, and $[T]$ is orthogonal, the

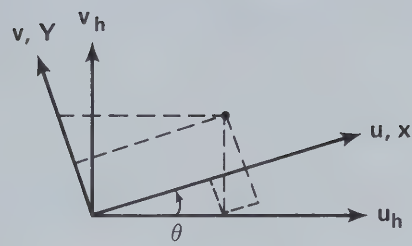


Figure 3.3 Orientation of Reference Frames

$$[\bar{T}]^T [\overset{*}{K}]_{lm} [\bar{T}] = \begin{bmatrix} C & -S & 0 \\ S & C & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} C & S & 0 \\ -S & C & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} [C^2K_{11} + S^2K_{22} - CS(K_{12} + K_{21})] & [CS(K_{11} - K_{22}) + C^2K_{12} - S^2K_{21}] & [CK_{13} - SK_{23}] \\ [CS(K_{11} - K_{22}) - S^2K_{12} + C^2K_{21}] & [S^2K_{11} + C^2K_{22} + CS(K_{21} + K_{12})] & [SK_{13} + CK_{23}] \\ [CK_{31} - SK_{32}] & [SK_{31} + CK_{32}] & [K_{33}] \end{bmatrix}$$

Figure 3.4 Transformed Submatrix $[Q]_{lm}$ (A plane frame problem)

product $[T]^T [K^*] [T]$ is also symmetric, and represents the orthogonal transformation of $[K^*]$ from the local reference frame to the higher level reference frame. Call the transformed stiffness matrix [Q]. If $[K^*]_{\ell m}$ is a qxq nodal partition of $[K^*]$, the corresponding partition of [Q], $[Q]_{\ell m}$, can be written as

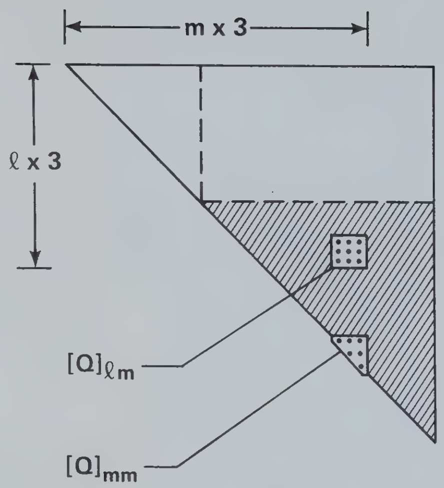
$$[Q]_{\ell m} = [\bar{T}]^T [K^*]_{\ell m} [\bar{T}] \tag{3.7}$$

Notice that $[K^*]_{\ell m}$ and $[Q]_{\ell m}$ are not symmetric if $\ell \neq m$. The explicit form of this transformed matrix $[Q]_{\ell m}$ is shown in Fig. 3.4.

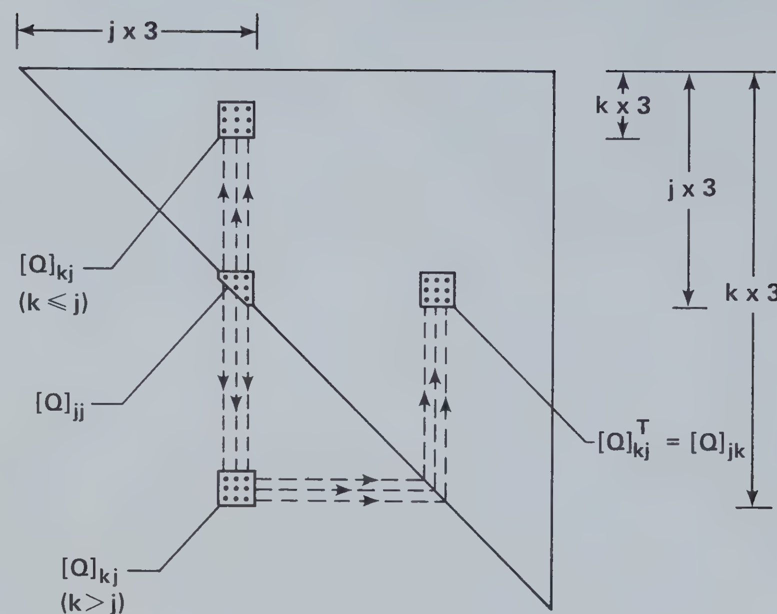
3.3 Assembly

Following the definition of [Q], the assembly of an inter-boundary partition in a higher level system matrix can be carried out in the same manner as the usual direct stiffness assembly. However, rather than assembling individual stiffness coefficients, qxq submatrices are assembled into qxq spaces in the higher level system stiffness matrix. In other words, instead of identifying the degrees of freedom of a substructure inter-boundary partition with the corresponding degrees of freedom of a higher level system, we need only identify the nodes of the inter-boundary partition with the nodes of the higher level system. This nodal connectivity information is very similar to the element nodal connectivity of simple direct stiffness techniques, and can be supplied either with every substructure, or with the higher level system data.

The assembly scheme is shown in Fig. 3.5. Some difficulties arise from the fact that we are dealing only with upper triangles of the matrices. For example, submatrix $[Q]_{\ell m}$ for a substructure system, defined



(a) Decomposed Substructure Stiffness Matrix (Shaded area is [K] *)



(b) Higher Level System Stiffness Matrix

Figure 3.5 The Assembly Scheme (Nodes l and m of the substructure identify with nodes k and j of the higher level system, respectively.)

by Eq. 3.7, may assemble into Q_{kj} in the higher level system, where nodes ℓ and m of the substructure identify with nodes k and j of the higher level system. Since we have only the upper triangle of the substructure stiffness matrix, $m \geq \ell$. If $j \geq k$, submatrix $[Q]_{kj}$ occupies a place inside the upper triangle of the higher level system matrix. If $j < k$, submatrix $[Q]_{kj}$ is outside the upper triangle. But, since the higher level system matrix must be symmetric, then $[Q]_{jk} = [Q]^T_{kj}$ is assembled rather than $[Q]_{kj}$. This procedure is shown in dotted lines in Fig. 3.5.

Following the first assumption of Section 3.1, when the first inter-boundary node in a substructure is identified, the higher numbered nodes are all inter-boundary nodes, and can be taken in order. The assembly then can proceed across the columns of the inter-boundary partition of $[K^*]$, q columns at a time.

In conclusion, it must be pointed out that the operations of transformation, according to Fig. 3.4, and assembly have been carried out simultaneously in order to reduce the operations needed to access the components of the qxq submatrices.

3.4 The Skyline of a Higher Level Unit

The assembly process described above controls the skyline of the higher level unit and access of $[Q_{kj}]$ depends on the location of $[Q_{jj}]$ which can be found from MAXA for the higher level unit. Establishing this skyline can take several forms. This will be discussed briefly in the following.

3.4.1 The Direct Approach

If the inter-boundary partition of a lower level unit is assumed fully populated, it can be considered as an element, and the process of computing and updating the column heights for the higher level unit would be the same as that described in Section 2.7.1. However, this approach is simplistic, since the assumption of fully populated inter-boundary partitions will result in more storage space than is actually required, as well as a definite increase in the number of numerical operations performed during assembly, decomposition, and backsubstitution. This approach is the one primarily used in the programs developed in Chapter 4, and forms the basis of the storage and CPU time analysis in Chapter 5. However, two alternative schemes to adjust the storage and access for the higher level unit to the 'exact' skyline requirement have been developed, as indicated below.

3.4.2 Skyline Modification

Using the direct approach described above, it is expected that for some columns, the actual first non-zero component will be somewhere down the column from the location indicated by the direct higher level assembly procedure. Thus, if this can be checked after assembly, the array MAXA modified accordingly, and the components of the stiffness storage vector shifted forward as may be necessary (Fig. 3.6), the true skylines of the lower level units would have been taken into consideration in an indirect manner. This approach will result in a possible lower skyline for the higher level unit, and a saving in the number of numerical operations required for decomposition and backsubstitution, but not in storage requirements, since this modification must necessarily

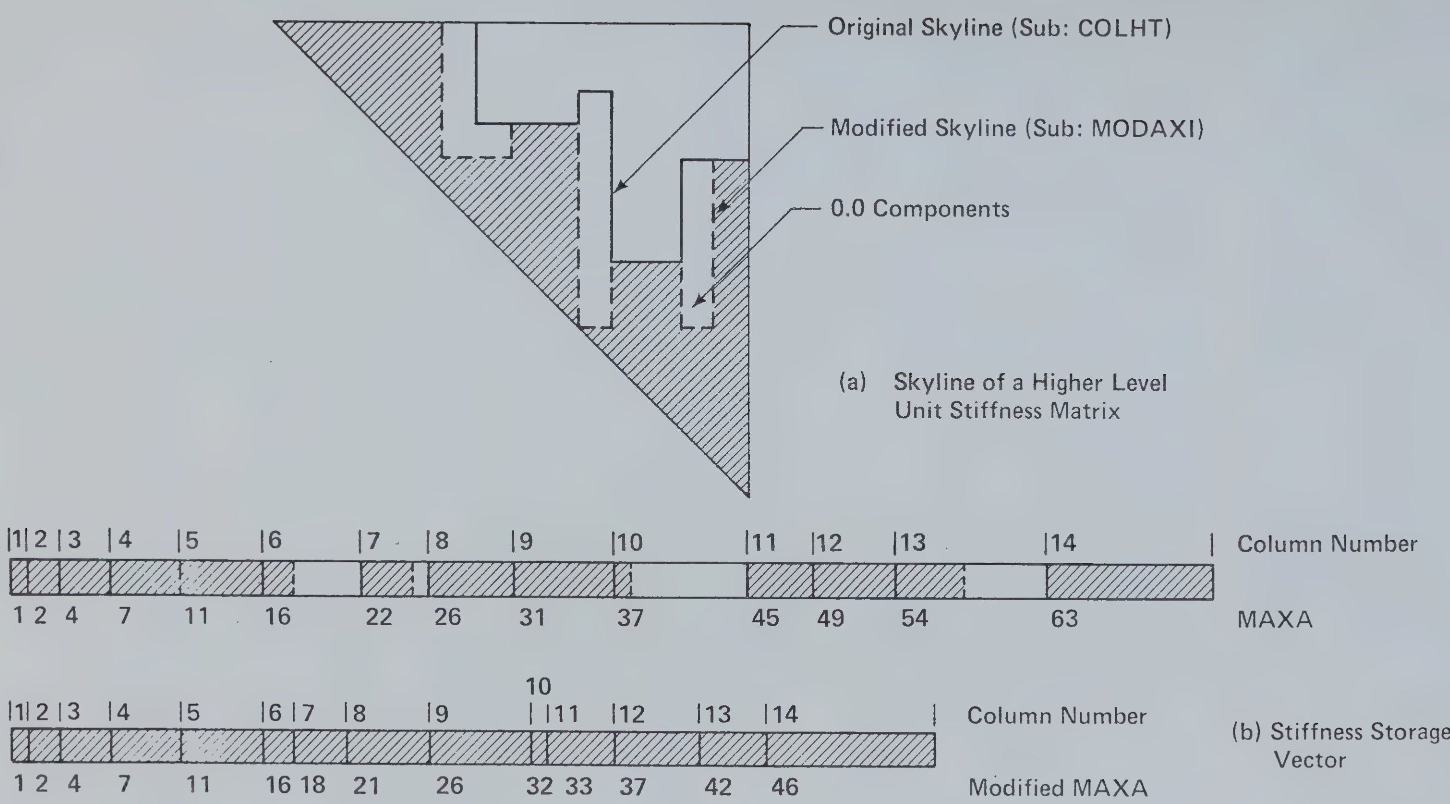


Figure 3.6 Skyline 'Modification 1' Scheme

take place after the assembly of the higher level unit stiffness matrix. This modification has been coded in subroutine MODAX1, see Appendix C, and the resulting saving in CPU time as applied to program MUSAPF, can be found in Table 5.2 denoted by 'modification 1'.

3.4.3 Predicting the Skyline

In this Section a rigorous approach to establishing the skyline of a higher level unit, taking into consideration the skylines of the lower level units, is developed. The scheme is very similar to the assembly scheme described in Section 3.3 and Fig. 3.5. Utilizing the two assumptions mentioned in Section 3.1, the following algorithm may be established.

For all the inter-boundary nodes (m) of the lower level unit taken in order, the following operations are performed.

1. For all submatrices $[Q_{\ell m}]$ of the lower level inter-boundary partition, starting at the main diagonal and ending at the level of partition or the skyline, whichever is lower, identifty nodes k and j in the higher level system corresponding to nodes ℓ and m respectively in the lower level system.
2. If $k \leq j$, the proposed column heights for the degrees of freedom of node j are functions of the difference (j-k). If these column heights are greater than the current column heights, update the latter.
3. If $k > j$, the proposed column heights for the degrees of freedom of node k are functions of the difference (k-j). If these column heights are greater than the current column heights, update the latter.

This approach establishes a skyline which is somewhere between the skylines determined by the approaches of Sections 3.4.1 and 3.4.2. It has been coded in subroutine SKYPRD, see Appendix C. The resulting storage requirements and CPU time can be found in Tables 5.1, and 5.2 as 'modification 2'.

3.5 External Boundary Conditions

As stated in Section 3.1, there will be a fixed number of degrees of freedom per node, and the local ordering of these degrees of freedom will be constant through the problem. The method of imposing the external boundary conditions on the stiffness matrix must not violate this assumption. The method used in programs SISAPF and MUSAPF is to attach a spring with a very high stiffness relative to the structural stiffness coefficients, to the node where a degree of freedom is restrained in the direction of this degree of freedom. This method, in addition to its simplicity allows for specifying non-zero displacements.

The coding of programs SISAPF and MUSAPF allows the user to impose the external boundary conditions at the basic substructure level and/or at a higher level of assembled substructures.

CHAPTER 4 - DEVELOPMENT OF PROGRAMS

4.1 Introduction to Program Development

Two substructure displacement analysis programs have been developed. Program SISAPF: Single-level Substructure Analysis for Plane Frames is a one level substructure analysis program, while MUSAPF: Multi-level Substructure Analysis for Plane Frames is a multi-level substructure analysis program. Both programs handle only plane frame type problems with no limitations on the size, number of substructures, and number of cases of loading.

The two programs use the equation solving package developed in Chapter 2 and described in Appendix C, as well as the coordinate transformation and assembly schemes described in Chapter 3. All subroutines developed are common to both programs, with the exception of the main executive routines and the input routines. A dry run facility is incorporated in both programs to check the input data, and to determine the size of storage required for the stiffness matrices.

The programs use a type of dynamic array allocation, proposed by McCormick [8]. This simple data manager identifies an array by its FORTRAN name, its length, and a logical number which designates the common block to which the array belongs. Further, the data manager can free the core storage assigned to the last defined arrays in any common block, or alternatively, it can free an entire common block.

Data transfer to and from backing storage is done using unformatted read and write statements on sequential files. To achieve a high degree of flexibility in program MUSAPF, two IBM system routines, NOTE and POINT, are used to move the read and write pointers freely over the

sequential files.

4.2 SISAPF: A Program for Single-level Substructure Analysis of Plane Frames

This program performs a one level substructure displacement analysis of plane frames. It is written for large problems which possess a degree of regularity in structural properties. This Section describes the organization and flow of SISAPF.

4.2.1 Concepts and General Description of SISAPF

The program is composed of two main phases. The first phase is user controlled and involves the definition and formulation of the problem. The second phase is the solution and output phase.

The definition of the problem has several aspects. A structure can be divided into a number of substructures, for which the inter-boundary connectivity and orientations can be considered global parameters, and do, in fact, define the master system. Yet, some of these substructures may be identical in structural properties relative to a local reference frame. If these identical recurrent substructures have the same number of inter-boundary nodes, they may be completely defined by specifying the structural properties of a representative structural unit (defining a substructure 'class'), together with the global substructure identification numbers of the individual substructures represented by this class.

The inter-boundary connectivity of these individual substructures describes the master system. This information is entered as array NPG.

Each column of this array holds the node numbers in the master system to which the substructure's interboundary nodes attach, starting with the first interboundary node of the substructure at the top of the column and taking the rest of these nodes in order. Table 4.1 describes array NPG for the master system of the example in Fig. 4.1.

Within a class of substructures the same loads may be imposed on a number of substructures. Instead of entering the loads of each individual substructure separately, only the independent cases of loading within a class of substructures are entered in one array, together with a load case identification array, KSUB, which assigns to each substructure a number of local cases of loading equal to the number of global cases of loading. These loads are referenced to the local reference frame. For example, the building shown in Fig. 4.1a can be divided into 5 substructures as shown in Fig. 4.1b. Substructures with global numbers 2, 3, and 4 have the same structural properties, and the same number of interboundary nodes. Thus they form a class of substructures, say class number 2. Substructures number 1 and 5 form substructure classes 1 and 3 respectively. This example has two cases of loading, shown in Fig. 4.1c, and it can be seen that substructure units 3 and 4 have identical loads for global load case 1, while substructure unit 2 has a different pattern. For global load case 2 all three substructures have identical loads. Thus inside substructure class number 2, there are three independent cases of loading that together describe all the global loads on all units inside this class. These three loading patterns can be entered in one array, together with the load case identification array KSUB shown in Table 4.2.

Since the technique of defining the connectivity and loading

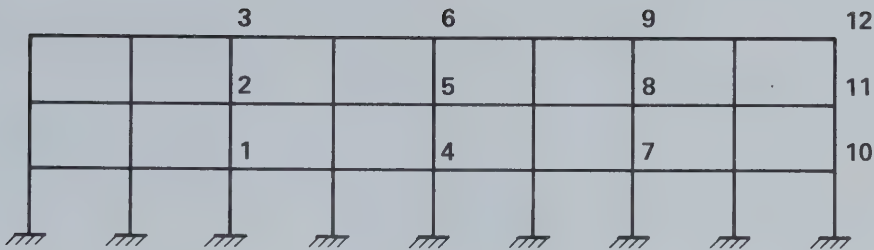


Figure 4.1a The Original Structure

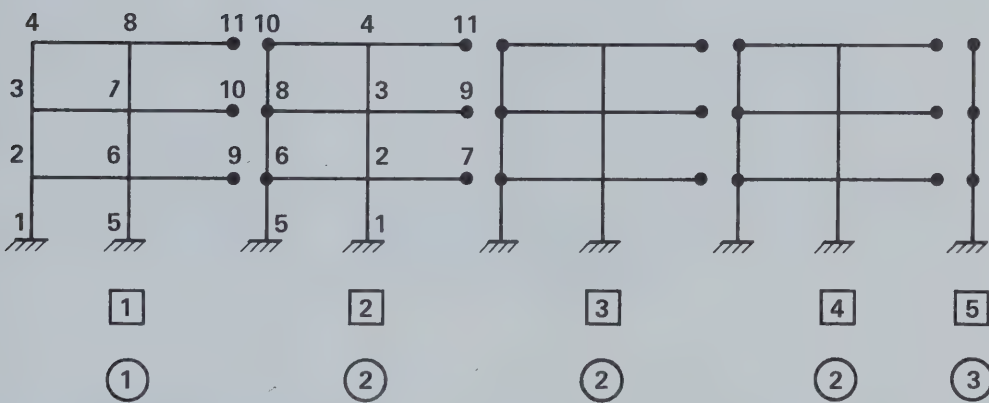


Figure 4.1b The Substructures

☐ Global Substructure Numbers ◯ Substructure Class Numbers,

● Inter-boundary Nodes

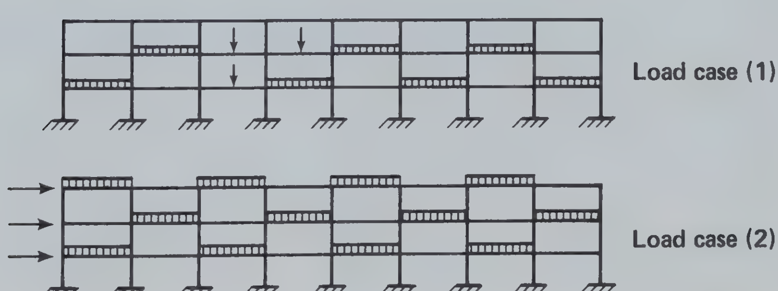


Figure 4.1c Global Load Cases on The Structure

Figure 4.1 Details of Example for SISAPF

Global Substructure Number	1	2	3	4	5
Substructure Class Number	1	2	2	2	3
Number of Inter-boundary Nodes	3	6	6	6	3
Array NPG	1	1	4	7	10
	2	4	7	10	11
	3	2	5	8	12
		5	8	11	
		3	6	9	
		6	9	12	

Table 4.1 Array NPG for Master System of Example in Figure 4.1

Local Substructure Number	1	2	3
Global Substructure Number (NGSUB)	2	3	4
Local Load Case Number for GLC. 1	1	2	2
Local Load Case Number for GLC. 2	3	3	3

Table 4.2 Arrays NGSUB and KSUB for Substructure Class 2 of Example in Figure 4.1

of a problem has now been described it is possible to return to a description of the program. The MAIN program segment is composed of: a number of common statements; a number of statements where the lengths of the common blocks are specified and passed to the data manager; a call statement to the main executive routine MAINMG; and, an END card. In this way the lengths of the common blocks can be exactly calculated prior to run time, and passed to the program through this small program segment which is compiled at run time. The program can be used to predict the required lengths of the common blocks, by employing the dry run facility described in Appendix A.

The executive routine MAINMG calls all other subroutines, which read the data, formulate and solve the problem, and output the final results.

4.2.2 Logic Flow of SISAPF

The sequence of events coded into the main executive routine MAINMG of program SISAPF is as follows. Details of the required input are given in Appendix A.

1. Read the problem control parameters (Sub: INPUT1).
2. If the problem is unsubstructured, go to 9.
3. Read the master system connectivity information, as well as substructure orientations (Sub: INPUT2).
4. Read the master system external boundary conditions, if any (Sub: BOUND).
5. Form the array of column heights for the master system stiffness matrix, by looping over the connectivity data supplied in step 3 (Sub: COLHT).

6. Form the addressing array for the diagonal components of the master stiffness matrix in the master system stiffness vector (Sub: ADDRES).
7. If this is a dry run, go to 9.
8. Reserve and clear space for master system stiffness and load vectors.
9. Set ICLSUB = 1, and start to loop over substructure classes, in order to read the data, form, reduce, and assemble the stiffness and load vectors.
10. Read substructure class ICLSUB control parameters (Sub: INPUT3).
11. Read nodal geometry, member material and cross section properties, member connectivity data, and arrays KSUB, and NGSUB for substructure class ICLSUB (Sub: INPUT4).
12. Read external boundary conditions, if any (Sub: BOUND).
13. Read nodal loads, and/or prescribed displacements (Sub: JLOAD).
14. Read member loads, and, if this is not a dry run, compute the corresponding nodal loads (Sub: MLOADS).
15. Form the column height array for the substructure class ICLSUB stiffness matrix (Sub: COLHT).
16. Form the diagonal component addressing array for the substructure class ICLSUB stiffness vector (Sub: ADDRES).
17. If this is a dry run, go to 23.
18. Form and assemble member stiffnesses into the substructure class ICLSUB stiffness vector (Sub: STIFF).
19. Add external boundary conditions if any to the substructure class ICLSUB stiffness vector (Sub: BOUND2).
20. If required, decompose the stiffness partition $[K_{ii}]$ and reduce the load partition $\{R_i\}$ (Sub: EQSBST) according to the algorithm of

Fig. 2.7. If the problem is unsubstructured, go to 28.

21. Reduce the stiffness partition $[K_{ib}]$ to $[F]^T$, and form the inter-boundary partitions $[K^*]$ and $\{R^*_b\}$ according to the algorithm in Figs. 2.8 and 2.9 (Sub: EQFT and Sub: EQKBB).
22. For each substructure that belongs to class ICLSUB, assemble partitions $[K^*]$ and $\{R^*_b\}$ into the master stiffness and load vectors, with the appropriate coordinate transformations (Sub: ASSEMB).
23. If the total number of classes of substructures equals 1, go to 31.
24. If this is a dry run, go to 26.
25. Store array pointers, control parameters, and all relevant arrays of substructure class ICLSUB on FILE 1.
26. Free common blocks SUBIA, SUBRA, and SUBSR.
27. If ICLSUB equals the total number of classes of substructures, go to 31. Otherwise, go to 10 to start on the next class of substructures.
28. This is a branch to an unsubstructured problem, reached from step 20. Backsubstitute from the decomposed stiffness matrix into the reduced load vector, to obtain the nodal displacements (Sub: BKSB1).
29. Output displacements and member end forces (Sub: DISPL and Sub: STRESS).
30. Return to MAIN. End of unsubstructured problem.
31. If this is a dry run, go to 42.
32. Add external boundary conditions, if any, to master system stiffness vector (Sub: BOUND2).
33. Solve for master system nodal displacements (Sub: EQSBST and Sub: BKSB1).

34. Start the backsubstitution and output control loop. Rewind FILE 1, and set ICLSUB = 1.
35. If the total number of classes of substructures equals 1, go to 37.
36. Read array pointers, control parameters, and all relevant arrays of substructure class ICLSUB from FILE 1.
37. Start to loop over all substructures that belong to class ICLSUB. The global identification numbers can be obtained from the array NGSUB. For each substructure, steps 38 to 40 will be performed.
38. Copy the inter-boundary nodal displacements from the master system solution vector into a dummy array BA, referenced to the local reference frame of this substructure. Also copy partition $\{R^*_i\}$ from the substructure class loading array into BA. The local load case number corresponding to any global load case can be obtained from array KSUB (Sub: RESUB).
39. Backsubstitute, if required, from the decomposed substructure class stiffness matrix into the dummy vector BA, to obtain the substructure internal nodal displacements (Sub: BKSB2, and Sub: BKSB1).
40. For each global load case, output the nodal displacements, and member end forces for this substructure.
41. Set ICLSUB = ICLSUB + 1. If ICLSUB is less than or equal to the total number of classes of substructures, go to 36 to start on the next class of substructres.
42. Return to MAIN. End of problem.

The function of subroutines INPUT1, INPUT2, INPUT3, and INPUT4 is described in steps 1, 3, 10 and 11 respectively. Subroutines

COLHT, ADDRES, EQSBST, EQKBB, EQFT, BKSB1, and BKSB2 form the equation solving package, and are listed in Appendix C. Subroutines BOUND, JLOAD, MLOADS, STIFF, BOUND2, ASSEMB, DISPL, STRESS, and RESUB are common to programs SISAPF and MUSAPF, and are listed in Appendix C. The data transfer to and from FILE 1 is carried out by four short subroutines, contained in Appendix C. Finally the data manager is also common to both programs and is listed in Appendix C.

4.3 MUSAPF: A Program for Multi-level Substructure Analysis of Plane Frames

This program is a more advanced version of program SISAPF, that deals with large complex problems that can be decomposed into a heirarchy of substructures. In its present form, it can handle only plane frame type problems. It has no limitations on the number of substructures or the number of levels of the heirarchy, and also has no limitations on the number of cases of loading. This Section describes the organization and flow of MUSAPF.

4.3.1 Conceptual Development

The concept of a heirarchy of substructures is outlined in Chapter 1. However, an illustrative example is useful to describe how the concept has been implemented. The illustrative structure shown in Fig. 4.2a can be partitioned into 9 basic structural units (Fig. 4.2b). Two overlapping numbering systems will be used to designate the various substructures. The first, a 'global numbering' system identifies each substructure as a physical component. In Fig. 4.2b, global units 1 and

9 have the same structural properties, and hence they form a class of substructures for which the structural configuration will be designated by 'independent basic unit' number 1. This independent basic unit will be denoted as IBU (1). Global unit numbers 2, 4, 6, and 8 have identical structural properties, which will be designated by IBU (2). In the same manner global unit numbers 3, 5, and 7 can be represented by IBU (3). Global units 1 and 2 can be assembled along their inter-boundary nodes, as shown in Fig. 4.2c. This new assembled unit is a higher level substructure unit. Give this unit global number 10, and since it is an independent structural entity, designate it IAU (4), which stands for 'independent assembled unit' number 4. In the same manner global unit pairs 3 and 4, 5 and 6, 7 and 8, can be assembled to form global units 11, 12, and 13, respectively. These three new higher level substructure units have identical structural properties, having been assembled from the inter-boundary systems of 3 identical pairs of independent units. All three may be represented by IAU (5). Further, global units 10, 11, 12, 13, and 9 may be assembled together along the interboundary nodes (Fig. 4.2d), giving a still higher level substructure unit, which has global number 14, and may be designated by IAU (6). This last assemblage is unique, and forms the top of the hierarchy of substructures, call it the master system.

The loads can be prescribed physically at the independent basic unit level. For example, for the loading situation shown in Fig. 4.3a. IBU (2) has three cases of loading as shown in Fig. 4.3c. These cases of loading are independent loading patterns, which are pooled in one array, and which can represent all loading combinations on global

units 2, 4, 6, and 8. Load combinations affecting independent assembled units can be represented by inter-boundary forces arising from the possible load combinations on lower level units. The loading cases at all levels can be traced without difficulty from Fig. 4.3.

The control arrays for an IAU are INDSUB, IRCSUB, IBNSUB, NPG, LCSUB, and ORINT. These arrays are described in detail in Appendix B. However, for IAU (5) of the example, Table 4.3 shows the first three of the arrays, Table 4.4 shows array NPG (the connectivity array), and Table 4.5 shows array LCSUB.

Once the master system has been assembled, its nodal displacements can be obtained, and the backsubstitution process is begun to solve successively for the nodal displacements of all substructures. This process can follow several paths, but the main feature is that a unique solution vector must be obtained for each global unit of the heirarchy. The problem is to minimize the data transfer to, and from backing storage. Since it is necessary for the user to specify the way the substructures are to be combined into higher level global units, it has been decided to let the backsubstitution process also be controlled by the user, since after going through the former process the user has all the information required to specify the latter process. Output can be initiated once a basic unit has been reached. Table 4.6 illustrates a user prepared complete backsubstitution process for the above example. Row 2 of this table holds the global unit number of a higher level unit for which a solution vector exists, while row 3 has the global number of a lower level unit for which a solution vector is sought. Row 4 supplies the independent unit number representing the lower level unit. Row 5 gives the position of lower level unit in the control arrays of the

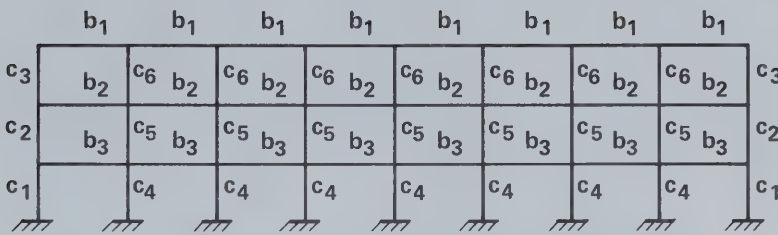


Figure 4.2a The Original Structure (c: column, b: beam).

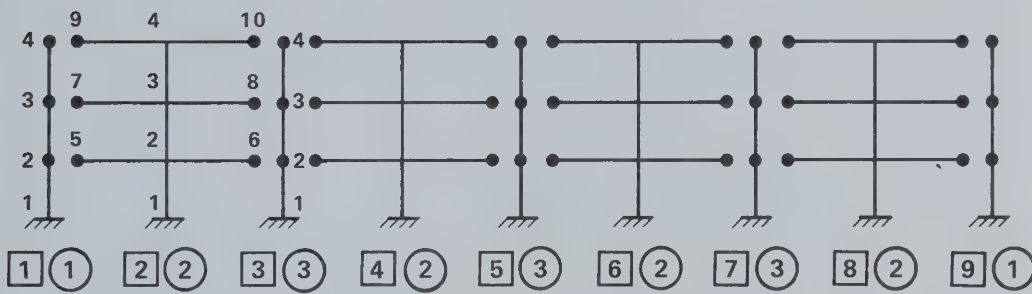


Figure 4.2b Basic Substructure Units

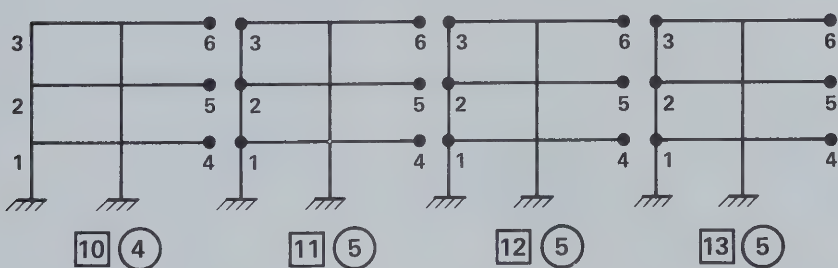


Figure 4.2c Assembled Substructure Units

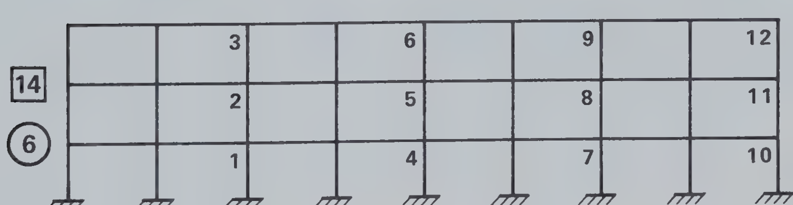


Figure 4.2d The Master System

Figure 4.2 A Hierarchy of Substructures

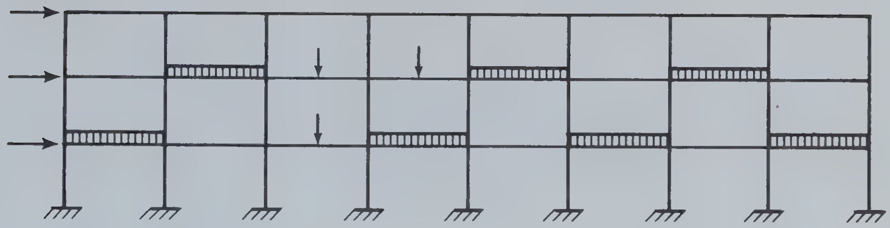


Figure 4.3a The Global Loading Pattern

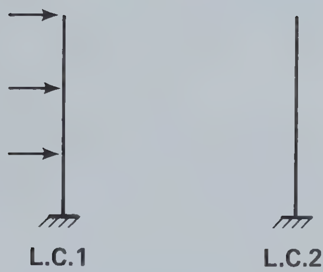


Figure 4.3b Loads on IBU(1)

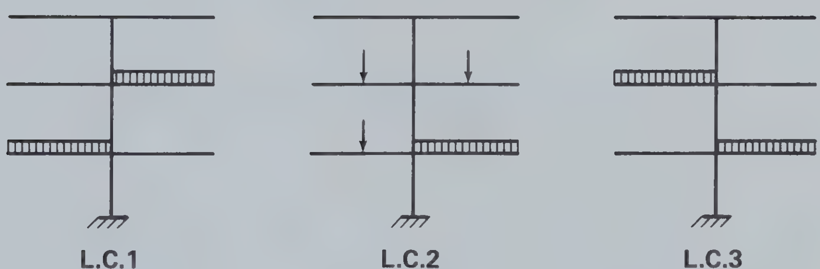
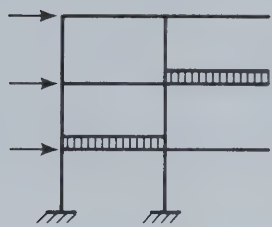


Figure 4.3c Loads on IBU(2)

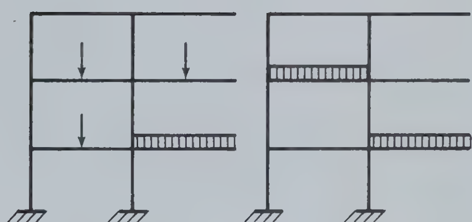
Figure 4.3d Loads on IBU(3)



L.C.1

Constituent Unit Number:	1	2
Load Case Numbers for L.C.1:	1	1

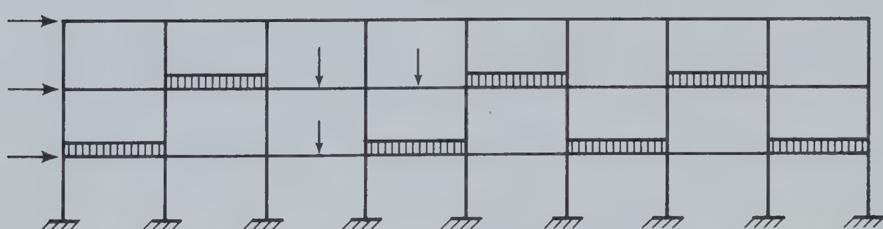
Figure 4.3e Loading Pattern on IAU(4)



L.C.1 L.C.2

Constituent Unit Numbers:	2	3
Load Case Numbers for L.C.1:	2	1
Load Case Numbers for L.C.2:	3	1

Figure 4.3f Loading Pattern on IAU(5)



L.C.1

Constituent Unit Numbers:	4	5	5	5	1
Load Case Numbers for L.C.1:	1	1	2	2	1

Figure 4.3g Loading Pattern on IAU(6)

Figure 4.3 Description of Hierarchial Loading

Constituent Unit Local Number	1	2	
Constituent Unit IBU(IAU) Number	3	2	INDSUB
Recurrence Flag for Constituent Unit	1	1	IRCSUB
Number of Inter-boundary Nodes	3	6	IBNSUB

Table 4.3 Arrays INDSUB, IRCSUB, IBNSUB for IAU(5)

Constituent Unit Local Number	1	2
	1	1
	2	4
	3	2
	-	5
	-	3
	-	6

Table 4.4 The Connectivity Array NPG for IAU(5)

Constituent Unit Local Number	1	2	
C.U. Load Case Number for IAU(5) L.C. 1	1	2	LCSUB
C.U. Load Case Number for IAU(5) L.C. 2	1	3	

Table 4.5 Array LCSUB for IAU(5)

Step Number	1	2	3	4	5	6	7	8	9	10	11	12	13
Higher Level Unit Global Number	14	14	14	14	14	10	10	11	11	12	12	13	13
Lower Level Unit Global Number	10	11	12	13	9	1	2	3	4	5	6	7	8
IBU or IAU of Lower Level Unit	4	5	5	5	1	1	2	3	2	3	2	3	2
Position of the IBU or IAU of the Lower Level Unit in the IAU Arrays of the Higher Level Unit.	1	2	3	4	5	1	2	2	1	2	1	2	1

Table 4.6 Backsubstitution Control

independent unit representing the higher level unit. For example, IAU (5) which represents unit 13 occupies position 4 in the control arrays of IAU (6) which represents unit 14, the 'master system'. This last parameter is necessary for the identification of the proper load cases of the independent lower level unit forming the global loading scheme of the lower level unit. The execution of Table 4.6 is carried out column by column. For example, executing the first column, the inter-boundary displacements of unit 10 are copied from the solution vector of unit 14, into a dummy array BA, referenced to the local reference system of unit 10. The proper load cases of IAU (4), representing the loads on unit 10 are obtained using the parameter of the last row, and the corresponding internal loading partitions are copied from the local load array of IAU (4) into the dummy array BA. Next, the stiffness matrix of IAU (4) is retrieved from backing storage, and backsubstitution to compute the internal nodal displacements of global unit 10 is carried out. As global unit 10 is not a basic substructure unit, array BA, the solution vector, will be stored on a file, and execution of the second column starts.

4.3.2 General Description of MUSAPF

The program uses the same data managing and equation solving packages as program SISAPF, as well as the same data transfer, formulation, and output subroutines. The MAIN program segment is similar to that of SISAPF. The difference between the two programs is that MUSAPF uses two types of substructures (basic units and assembled units), and also backsubstitution is a user controlled process in MUSAPF, as described in Section 4.3.1. The higher level control arrays are ORINT for orienta-

tion, and NPG for connectivity of the constituent units, as well as LCSUB which identifies the local load cases on a constituent unit which defines its effect on the higher level unit. The input description is found in Appendix B, and the listing in Appendix C.

4.3.3 Logic Flow of MUSAPF

The sequence of events coded into the main executive routine MAINMG of program MUSAPF is as follows.

1. Read problem control parameters (Sub: INPUT1).
2. Read backsubstitution control arrays (Sub: INPUT2).
3. Set ICLSUB = 1, and start to loop over the independent basic substructures units (IBU's), to read data, and form, reduce, and store the IBU stiffness and load vectors.
4. Read IBU(ICLSUB) control parameters (Sub: INPUT3).
5. Read IBU(ICLSUB) nodal geometry, member material and cross section properties, and member connectivity data (Sub: INPUT 4).
6. Read IBU(ICLSUB) external boundary conditions, if any (Sub: BOUND).
7. Read IBU(ICLSUB) nodal loads, and/or prescribed displacements, if any (Sub: JLOAD).
8. Read IBU(ICLSUB) member loads, and if this is not a dry run, compute member end forces (Sub: MLOADS).
9. Form the column height array for this IBU stiffness matrix (Sub: COLHT).
10. Form the diagonal component addressing array for this IBU stiffness vector (Sub: ADDRES).
11. If this is a dry run, go to 18.

12. Reserve and clear space for this IBU stiffness vector.
13. Form and assemble member stiffnesses into the IBU stiffness vector (Sub: STIFF).
14. Add external boundary conditions, if any, to this vector (Sub: BOUND2).
15. If required, decompose the stiffness partition $[K_{ii}]$ and reduce load partition $\{R_i\}$ according to algorithm of Fig. 2.7 (Sub: EQSBST).
16. If this is an unsubstructured problem, go to 19. Otherwise, reduce the stiffness partition $[K_{ib}]$ to the form $[F]^T$ according to the algorithm of Fig. 2.8, and form the inter-boundary partitions $[K^*]$,, and $\{R^*_b\}$ according to the algorithm of Fig. 2.9 (Sub: EQFT, and Sub: EQKBB).
17. Store the present values of the 'write' pointers of FILE 1 and 2 in arrays IRTRV1 and IRTRV2, respectively. Then store the array pointers, control parameters, and all relevant arrays of IBU(ICLSUB) on FILES 1 and 2.
18. Set ICLSUB = ICLSUB + 1, and if it is greater than the total number of independent basic units, go to 22. If not, go to step 4 to start on the next IBU.
19. This is a branch to an unsubstructured problem. Backsubstitute from the decomposed stiffness matrix, into the reduced load vector, to obtain the nodal displacements (Sub: BKSB1).
20. Output the nodal displacements and member end forces (Sub: DISPL, and Sub: STRESS).
21. Return to MAIN. End of unsubstructured problem.

22. Start to loop over independent higher level units. These are assemblies of more than one IBU or IAU unit. Their numbering system is a continuation of the numbering of the IBU's.
23. Read IAU(ICLSUB) control parameters (Sub: INPUT5).
24. Read the constituent units' identification numbers, connectivity and load case identification arrays (Sub: INPUT6). These constituent units may be IBU's or IAU's provided that they have been previously defined.
25. Read IAU(ICLSUB) external boundary conditions, if any (Sub: BOUND).
26. Form the column height array for the IAU(ICLSUB) stiffness matrix, by looping over the connectivity data supplied in step 23 (Sub: COLHT).
27. Form the diagonal component addressing array for the IAU(ICLSUB) stiffness vector (Sub: ADDRES).
28. If this is a dry run, go to 36.
29. Reserve and clear space for the IAU(ICLSUB) stiffness and load vectors.
30. Assemble the constituent units', inter-boundary stiffness and load partitions, into the IAU(ICLSUB) stiffness and load vectors with the appropriate transformations (Sub: ASSEMB). The constituent unit matrices can be obtained from FILE 2, for which the 'read' pointers are obtained from array IRTRV2.
31. Add external boundary conditions, if any, to the IAU(ICLSUB) stiffness vector (Sub: BOUND2).
32. If required, decompose the stiffness partition $[K_{ii}]$ and reduce the load partition $\{R_i\}$ according to the algorithm of Fig. 2.7 (Sub: EQSBST).

33. If this is the master system, go to 38.
34. Reduce the stiffness partition $[K_{ib}]$ to the form $[F]^T$ and form the interboundary partitions $[K^*]$, and $\{R^*_b\}$ according to the algorithms of Figs. 2.8 and 2.9, respectively (Sub: EQFT, and Sub: EQKBB).
35. Store the present values of the 'write' pointers for FILES 1 and 2 in arrays IRTRV1, and IRTRV2 respectively, and store the array pointers, control parameters, and all relevant arrays of IAU(ICLSUB) in FILES 1 and 2.
36. Free common blocks from this IAU's arrays.
37. If this is a dry run, and if this is the master system, go to 41.
38. Set ICLSUB = ICLSUB + 1. Go to 23 to read the data, and defint the next IAU.
39. Obtain the solution vector for the master system (Sub: BKSB1), and if required, store this vector in FILE 3.
40. Start the backsubstitution and output control loop. This process is an execution of a table similar to Table 4.6, which is supplied in step 2. If required, solution vectors can be stored on FILE 3, for which the write pointers are stored in array IRTRV3.
41. End of problem. Return to MAIN.

The input subroutines INPUT1 to INPUT6 are briefly described in steps 1, 2, 4, 5, 23, and 24, and they are not to be confused with subroutines INPUT1 to INPUT4 of program SISAPF.

CHAPTER 5 - APPLICATION

5.1 Description of Structure

A flat slab high rise structure [7] is analysed, using SISAPF and MUSAPF, both as a one unit structure and as a partitioned structure in several ways. An analysis of the execution times for several schemes used to partition the building is attempted.

In Figs. 5.1a and 5.1b, the equivalent frame [6] and a half plan of the building are shown. The columns and shear wall are of constant section and the slab is seven and a half inches thick. All sections are assumed uncracked. The link members are of high axial stiffness to simulate rigid floor diaphragms. The shear wall is approximated by a vertical member of the correct stiffness, and horizontal rigid projecting beams are used to simulate wide column to beam connections [14].

The structure is analysed under a lateral wind load of 20 psf intensity in the Y direction, see Fig. 5.1b. These loads are applied to the joints along the left side of the lumped frames and on the left side of the shear wall-frame.

5.2 Partitioning Schemes

Five different schemes have been used to partition the structure. The first scheme is an analysis of the structure as one unit and will be referred to as Example 1 when run on MUSAPF, and Example 6 when run on SISAPF. The second scheme is a single level substructure scheme, Fig. 5.2, and will be referred to as Example 2 when run on MUSAPF, and as Example 7 when run on SISAPF. In this scheme, there are 15 basic

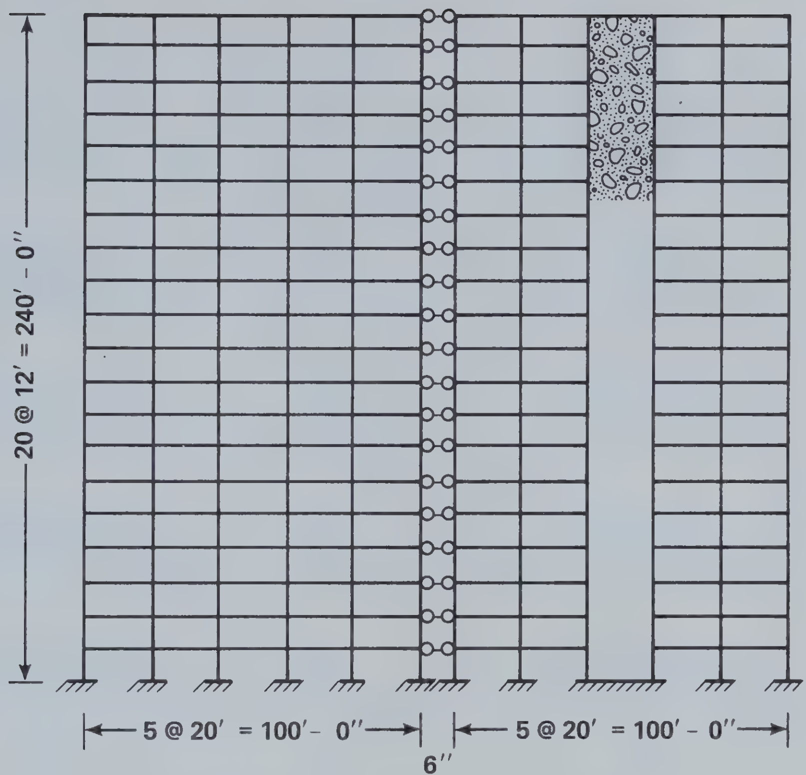


Figure 5.1a Elevation of frame

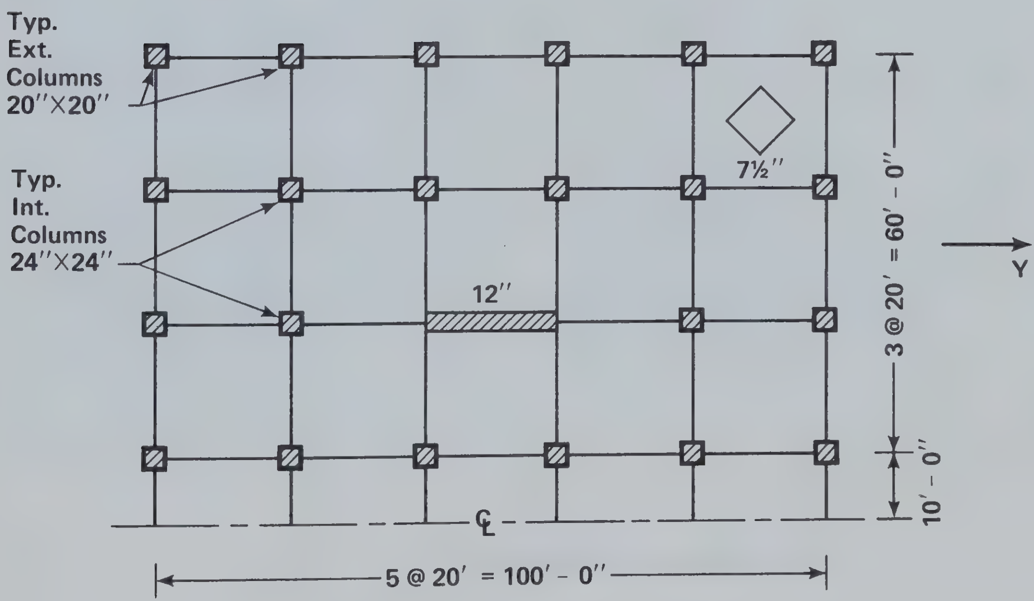


Figure 5.1b Typical Floor Plan of Structure

Figure 5.1 Details of Problem Structure

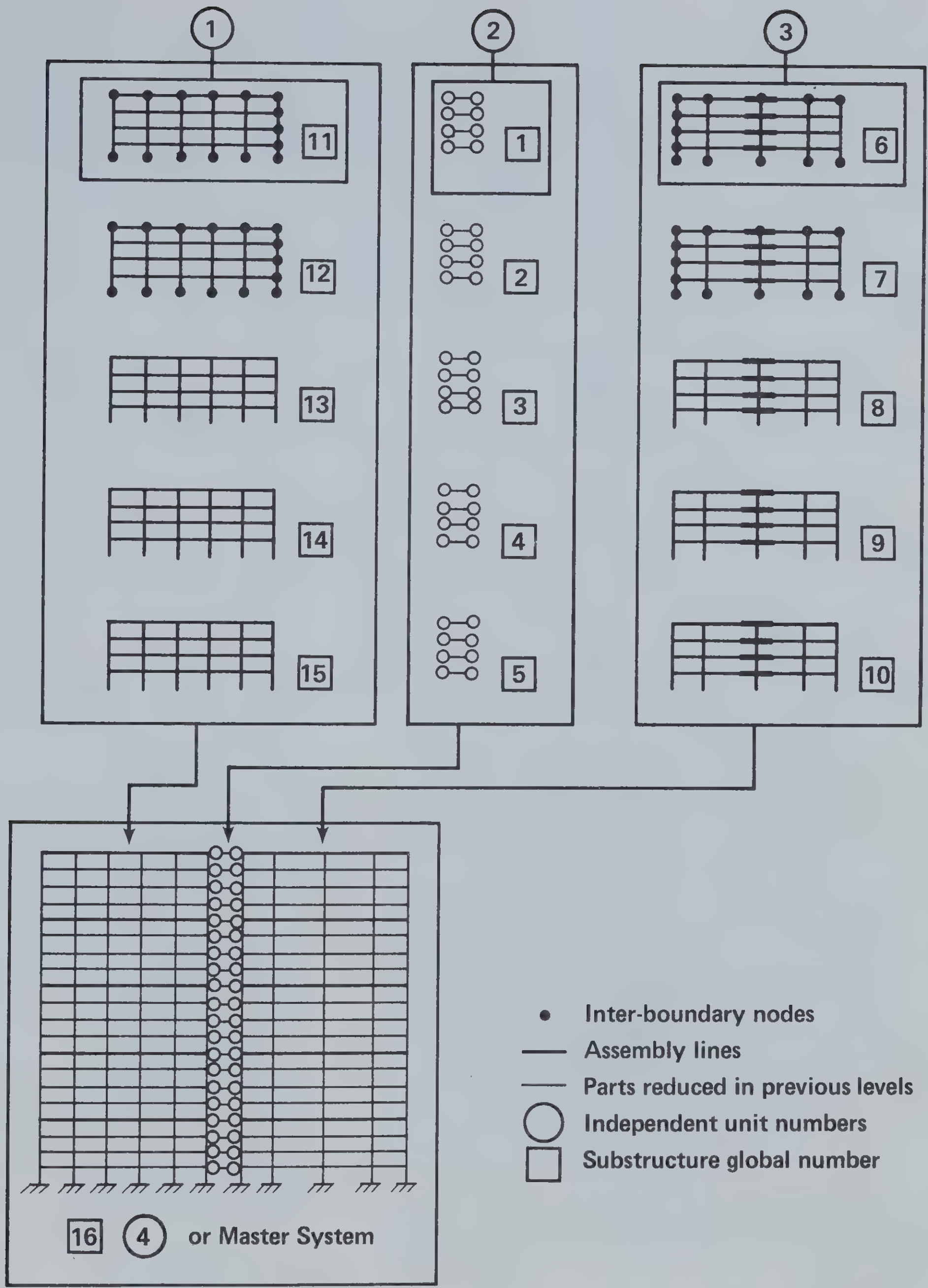


Figure 5.2 Substructure Scheme for Examples 2 and 7

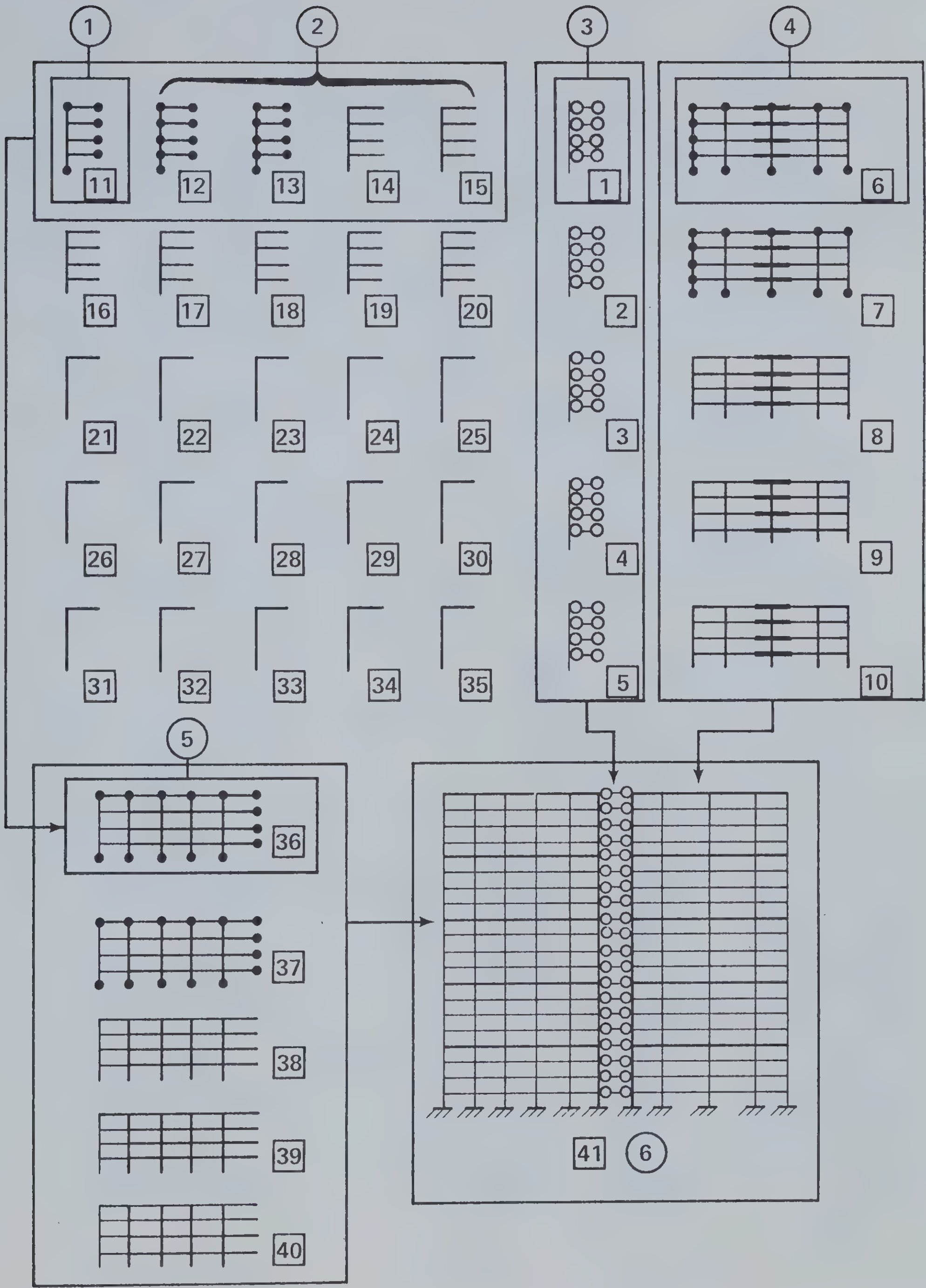


Figure 5.3 Substructure Scheme for Example 3

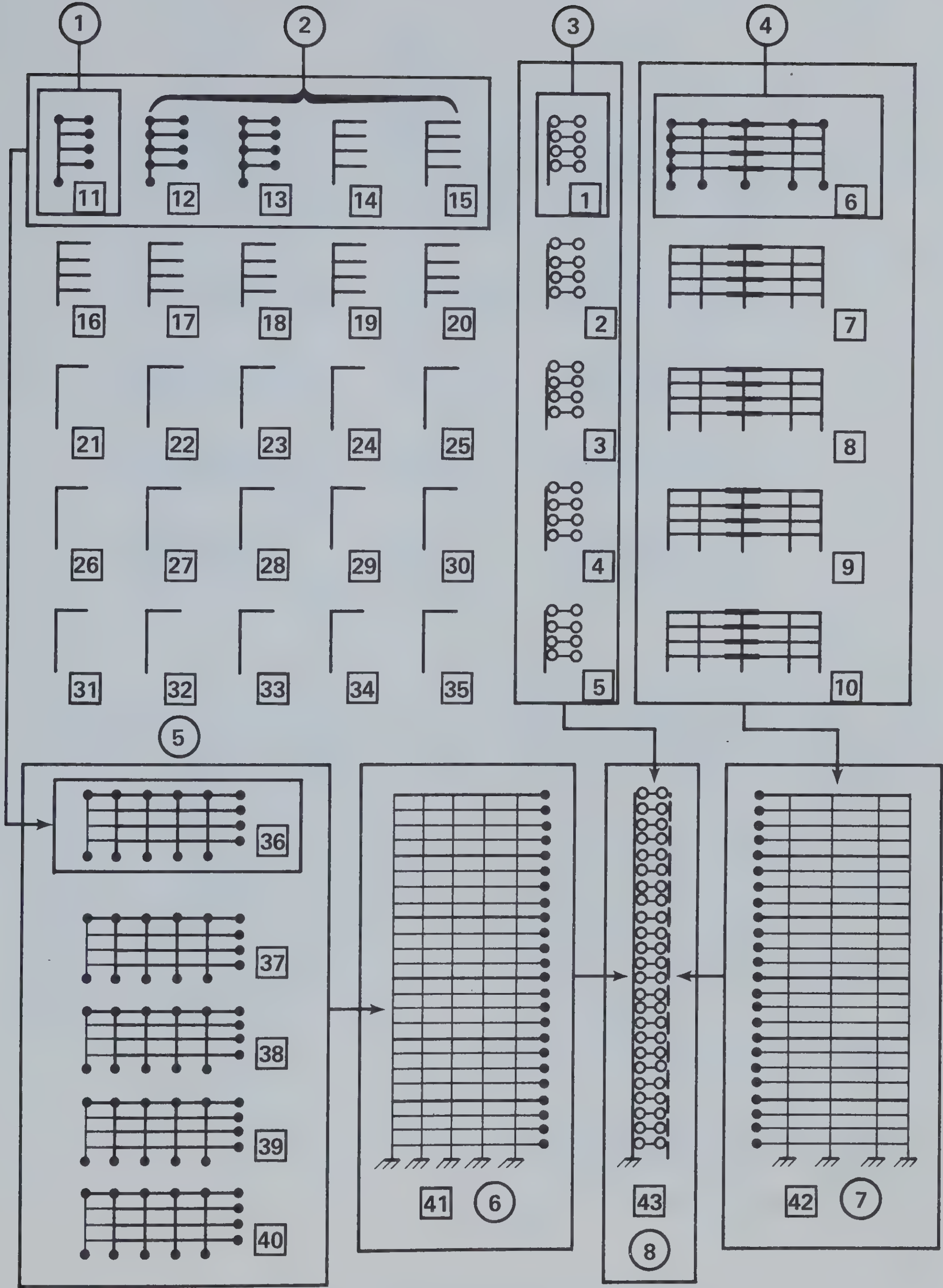


Figure 5.4 Substructure Scheme for Example 4

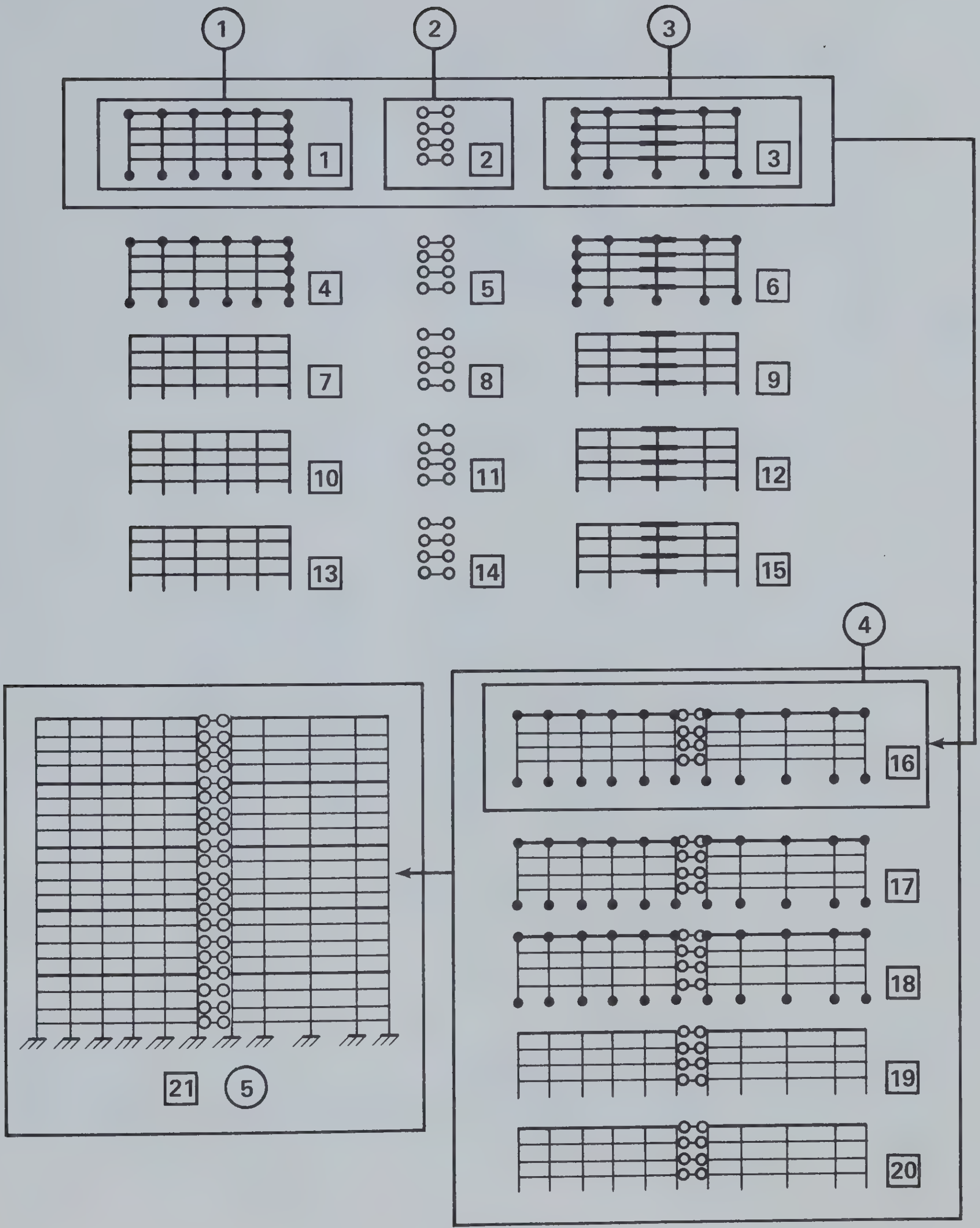
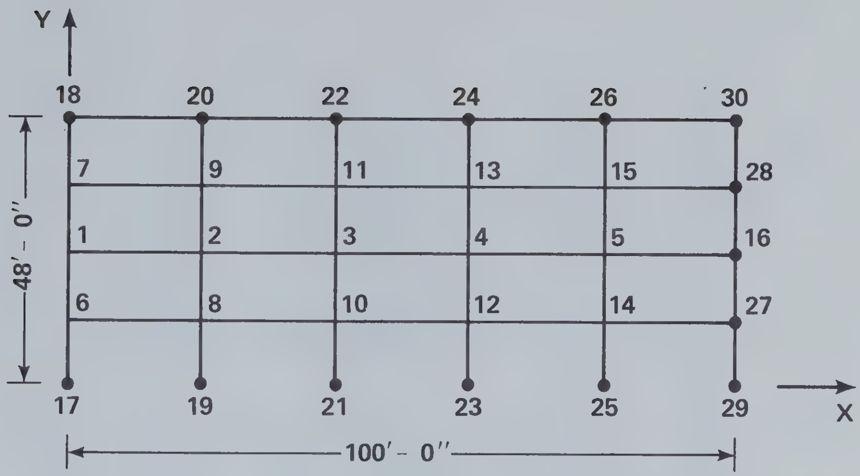
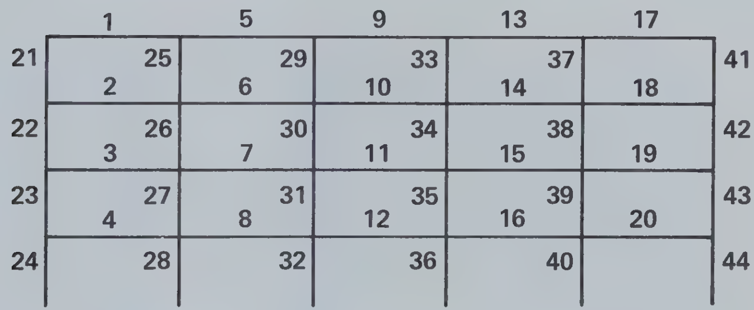


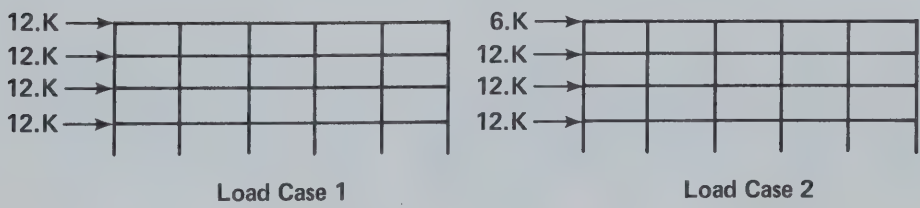
Figure 5.5 Substructure Scheme for Example 5



(a) Nodal Numbering and Geometry (● Inter-boundary node)

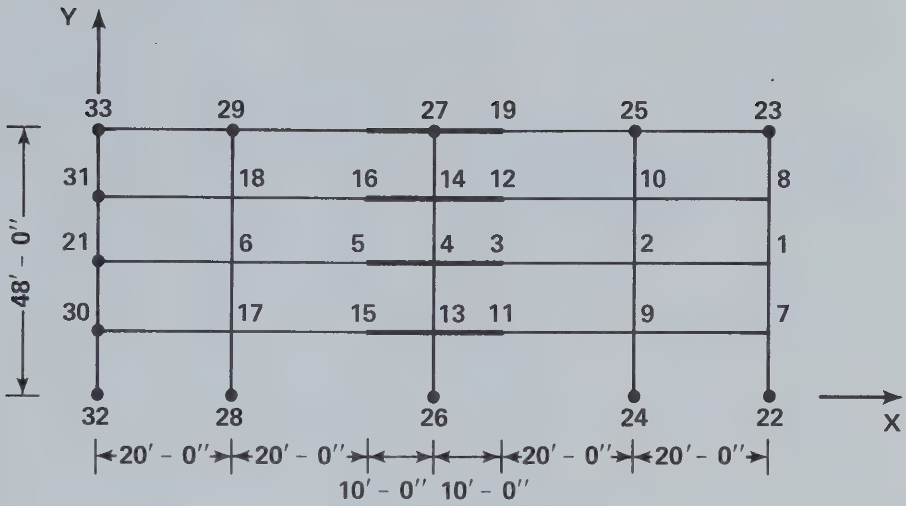


(b) Member Numbering

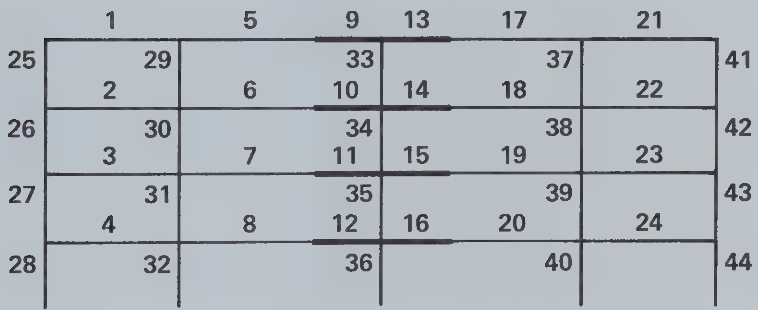


(c) Load Cases

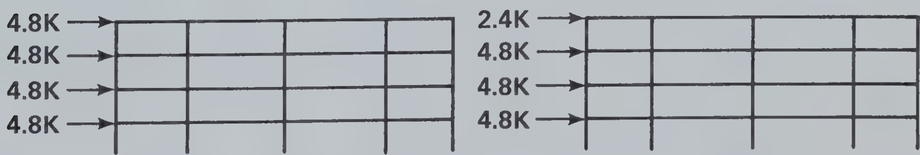
Figure 5.6 Details of IBU(1) for Examples 2 and 7



(a) Nodal Numbering and Geometry (● Inter-boundary nodes)



(b) Member Numbering



Load Case 1

Load Case 2

(c) Loads

Figure 5.7 Details of IBU(3) for Examples 2 and 7

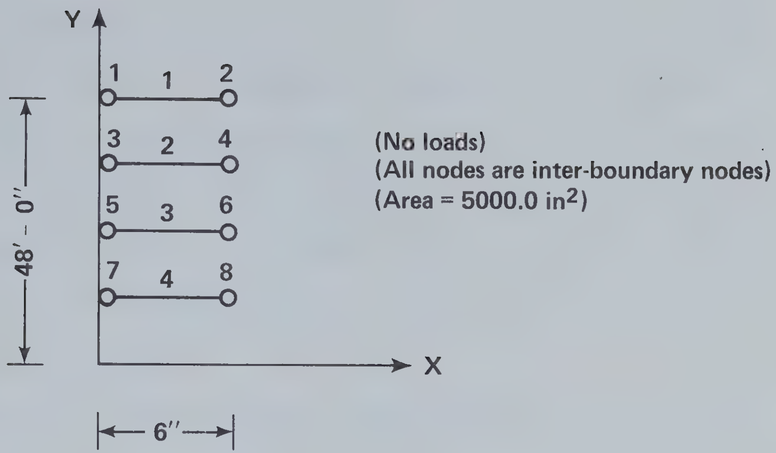


Figure 5.8 Details of IBU(2) for Examples 2 and 7

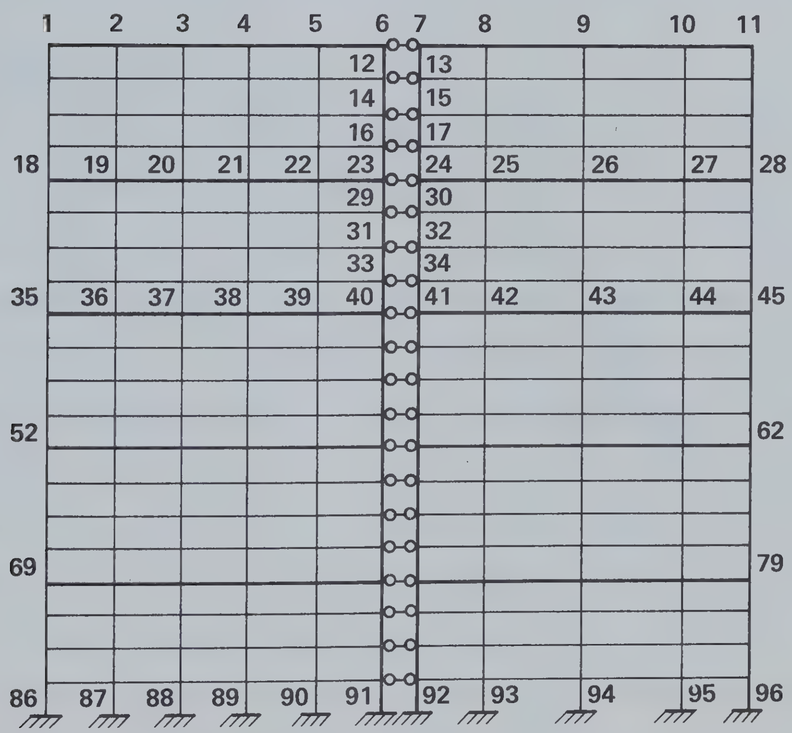


Figure 5.9 Details of IAU(4) or the Master System for Examples 2 and 7

substructure units divided into three classes, and one assembled unit which is the master system. The details of these units are shown in Figs. 5.6, 5.7 and 5.8, the master system is shown in Fig. 5.9, and the input files for both programs are in Appendix D. The third, fourth, and fifth schemes are multi-level substructure schemes, see Figs. 5.3, 5.4, and 5.5, respectively, for which the details have not been provided. These have been run on MUSAPF.

The analysis of all examples yielded identical displacements and member end forces for all nodes and members up to 6 decimal places which proves the accuracy of the equation solver, and the fact that the multi-level substructure scheme used, does not affect the accuracy.

5.3 Size and Core Storage

One of the main advantages of the substructure method is the saving in core space required to process a problem. Table 5.1 gives the control parameters, the master system stiffness size, and the maximum size of core space for each example in normal length words. It can be seen from Examples 1, 2, 3, and 4, that as the number of levels of substructures increases, the core space required decreases. The maximum core space required for any substructure scheme is between 35% and 47% of the size required for the structure as one unit. As indicated in Section 3.4, this advantage could be further enhanced, if the skylines of the lower level matrices were considered when computing the skylines of the higher level unit stiffness matrices. The discussion above is based on the 'direct assembly' approach, described in Section 3.4.1. The effects of modifying the skylines, using the assembly schemes described in Sections 3.4.2 and 3.4.3 are discussed in Section 5.6.

Program	MUSAPF					SISAPF	
Example	1	2	3	4	5	6	7
Total Number of Basic Units	1	15	35	35	15	1	15
Total Number of IBU's	1	3	4	4	3	1	3
Total Number of Units	1	16	41	43	21	1	16
Total Number of IAU's	-	1	2	4	2	-	1
Number of Levels	-	1	2	3	2	-	1
Maximum NWA (Locations)	27573	2340	2340	2340	2340	27573	2340
NWK (Locations)	-	13320	13095	7626	8811	-	13320
Maximum Core Space	76901	34909	34484	32737	27260	76322	35753
NWK (Modification 1)	-	11722	11723	5055	8811		
NWK (Modification 2)	-	12672	12258	5745	8811		

Notation: IBU: Independent Basic Unit
IAU: Independent Assembled Unit
NWA: Stiffness Vector Size for an IBU
NWK: Stiffness Vector Size for the Master System.

Table 5.1 Control and Size Parameters for Examples 1 to 7

5.4 CPU Time

Programs MUSAPF and SISAPF use the IBM system subroutine TIME to output the CPU time at every stage of execution. The values obtained depend to a certain degree on random machine conditions. However, a comparison is possible, if all examples are run in one session, taking care to specify the correct file sizes.

The stages of execution are, mainly: the reading and formulation, the decomposition, the storage on peripheral devices, and the backsubstitution and output. The CPU time taken by each stage is shown in Table 5.2 for all examples, as well as, the total CPU time.

It can be seen that all substructured examples, with the exception of Example 4, take less CPU time than the unsubstructured examples. The CPU time saving should increase if the element type requires large formulation time. The formulation time is an average of 15% of the total time for the substructured examples as opposed to 50% for the decomposition stage. The time required for backsubstitution is an average of 35% of the total time for the substructured problems. The backsubstitution time increases as the number of levels of substructures increases, whereas the decomposition time depends mainly on the size and number of the stiffness matrices.

5.5 Substructuring, and Nodal Numbering

In all examples, the nodal numbering, for both the basic level substructures and the assembled units, has been chosen carefully. The governing factor was the element connectivity for the former, and the constituent unit connectivity for the latter. Over a number of experi-

mental runs not included in those examples, it has been observed that numbering the master system vertically instead of horizontally as in Fig. 5.9, raises the decomposition CPU time by as much as 500%.

Partitioning a structure is better carried out parallel to the shorter side. This way the master system will have fewer nodes, and any particular substructure will span fewer interboundary nodes, resulting in higher efficiency. The reason is the fact that the master system nodes are necessarily inter-boundary nodes in some or all substructure units at all levels. The repeated assembly of, and backsubstitution through these nodes consumes more time as their number increases.

Numbering the inter-boundary nodes of any substructure, or the nodes of the master system parallel to the shorter side lowers the skyline of this particular system in the regions $[K_{ib}]$, and $[K_{bb}]$. Lowering the skyline means less storage and less CPU time.

5.6 Higher Level Unit Modified Skylines

The modifications to skylines of higher level units discussed in Section 3.4, have been applied experimentally to program MUSAPF. The approach characterized by checking the columns of an assembled higher level unit stiffness matrix for the actual first non-zero component, is referred to as 'modification 1', while the approach characterized by predicting the skyline of a higher level unit taking into consideration the skylines of the lower level units, is referred to as 'modification 2'.

Table 5.1 shows the change in stiffness storage vector length for modifications 1 and 2, for Examples 2 to 5. It must be noted that while modification 1 indicates a smaller storage requirement, this is

Program	MUSAPF					SISAPF	
Example	1[a]	2[b]	3	4	5	6[a]	7[b]
Reading and Formulation	0.780	0.473	0.485	0.666	0.527	0.731	1.024
Decomposition	1.906	1.474	1.468	2.724	1.294	1.921	1.310
Storage	-	0.034	0.039	0.091	0.969	-	0.034
Backsubstitution, Output	0.747	0.938	1.170	1.231	1.118	0.696	0.493
Total CPU Time	3.433	2.964	3.219	4.772	3.037	3.348	2.918
T. CPU time (Modification 1)	-	2.787	3.010	4.085	2.940		
T. CPU time (Modification 2)	-	2.919	3.074	4.187	3.050		

Notes: a: The structure is treated as one unit.
b: Single-level substructure schemes.

Table 5.2 CPU Time Consumption

not true in practice, since the modification takes place after the assembly. However the storage saving for modification 2 is real.

Table 5.2 compares the total CPU time consumed for Examples 2 to 5, for both modifications with the unmodified scheme. For 'modification 1', the saving varies between 14.4% for Example 4 to 3% for Example 5, while for 'modification 2', the saving varies between -0.4% for Example 5 to 12.3% for Example 4. Two trends can be observed; the first being the increased saving in CPU time as the number of levels of substructures increases, while the second is that as the nodal numbering approaches an optimum state, no significant change is observed.

'Modification 2' is more elaborate, and has a rigorous theoretical background compared to 'modification 1'. However, the latter appears to be more economical for several reasons. The number of numerical operations involved in 'modification 1' can never be greater than that of 'modification 2', and while the latter requires a certain amount of data retrieval, the former does not. Also 'modification 1' gives a better upper bound for the skylines as can be seen from Table 5.1, since it checks for the actual first non-zero component in a column, while 'modification 2' anticipates this component. The above reasoning is supported by the fact that 'modification 1' gives consistently less CPU time, and should be recommended as a permanent addition to programs MUSAPF, and SISAPF.

CHAPTER 6 - SUMMARY AND CONCLUSIONS

Two plane frame substructure analysis programs, SISAPF and MUSAPF have been developed. SISAPF is based on a single-level substructure scheme, and MUSAPF is based on a multi-level substructure scheme. For this purpose, an equation solving package based on the skyline technique utilizing the concept of 'Cholesky' decomposition, as well as an assembly and a coordinate transformation scheme have been developed.

From studies on a sample structure, it has been observed that using a substructure scheme the core space requirements can be reduced to 35% of that required for the same structure as one unit. A saving in CPU time amounting to 15% can be easily achieved. The saving in CPU time should be greater if more complicated structural elements are used since the saving in formulation time averages 37%. While the saving in decomposition time averages 26%, a rise in backsubstitution CPU time is observed and increases as the number of, and number of levels of, substructures increases.

Partitioning and inter-boundary node numbering are found to give best results when both are carried out parallel to the shorter side of the structure or substructure.

As they stand, both programs accept only one type of element. However, both can be developed further to accommodate structures where the number of nodes per element, and the number of degrees of freedom per node can be varied. Generalization to accommodate such changes would require modification of the assembly and coordinate transformation schemes.

The basic assembly scheme assumes that the inter-boundary

partitions of lower level substructure units are fully populated when assembling a higher level unit. This scheme has been improved by recognizing the skylines of the lower level unit stiffness matrices when assembling the higher level units, and developing two modified assembly schemes which take this into account.

REFERENCES

1. Bathe, K.J., and Wilson, E.L., 'NONSAP - A Nonlinear Structural Analysis Program', Nuclear Engineering and Design 29 (1974), pp. 266-293.

2. Bathe, K.J., and Wilson, E.L., "Numerical Methods in Finite Element Analysis', Prentice Hall Inc., Englewood Cliffs, New Jersey, 1976.

3. Felippa, C.A., 'Solution of Linear Equations with Skyline-Stored Symmetric Matrix', Lockheed Palo Alto Research Laboratory, Palo Alto, California, 1973.

4. Furuike, T., 'Computerized Multi-level Substructuring Analysis', Computers and Structures, Vol. 2, pp. 1063-1073, Pergamon Press, Great Britain, 1973.

5. Hrabock, M.M., 'The Method of Substructures and its Application to Structural Analysis', M.Sc. Thesis, University of Saskatchewan, Regina, Saskatchewan, 1975.

6. Khan, R.F., and Sbarounis, J.A., 'Interaction of Shear Walls and Frames', Journal of the Structural Division, ASCE, Vol. 90, No. ST3, June, 1964.

7. Mathews, G.S., 'The Second-Order Analysis and Design of Reinforced Concrete Frames', M.Sc. Thesis, University of Alberta, Edmonton, Alberta, 1977.

8. McCormick, J.M. 'Data Handling in FORTRAN Programs', Committee on Computers and Information Systems, American Society of Civil Engineers, National Structural Engineering Convention, 975.

9. Meyer, C., 'Solution of Linear Equations - State of the Art', Journal of the Structural Division, ASCE, Vol. 101, No. ST4, April 1975, pp. 869-890.

10. Murray, D.W., 'PLFRAM: Program for Plane Frame Direct Stiffness Analysis', University of Alberta, Edmonton, Alberta, 1975.

11. Przemieniecki, J.S., 'Theory of Matrix Structural Analysis', McGraw-Hill, New York, New York, 1968.

12. Rosen, R., and Rubenstein, M.F., 'Substructure Analysis by Matrix Decomposition', Journal of the Structural Division, ASCE, Vol. 96, No. ST3, March ,1970, pp. 663-670.

13. Schrem, E., 'Computer Implementation of the Finite Element Procedure', Numerical and Computer Methods in Structural Mechanics, Fenves, S.J., Perrone, N., Robinson, A.R., and Schnobrich, W.C., Ed., Academic Press, Inc., New York, New York, 1973, pp. 79-121.

14. Schwaighofer, J., and Microys, H.F., 'Analysis of Shear Walls Using Standard Computer Program', ACI Journal, December 1969.

15. Williams, F.W., 'Comparison Between Sparse Stiffness Matrix and Substructure Methods', International Journal for Numerical Methods in Engineering, Vol. 5, 1973, pp. 383-394.

APPENDIX A

USER'S MANUAL FOR SISAPF

APPENDIX A - USER'S MANUAL FOR SISAPF

A.1 Master System Data Cards

A.1.1 Heading Card (20A4)

One card which contains any title for the problem.

A.1.2 Master System Control Card (8I4, F12.0, I4)

4	8	12	16	20	24	28	32	44	48
NSUB	NCLSUB	MXND	NUMNST	NGLC	IDOF	NODE	MBEL	UNIT	IDRY

NSUB : Total number of substructures.

NCLSUB : Total number of classes of substructures.

MXND : Maximum number of inter-boundary nodes in any substructure.

NUMNST : Total number of nodes in the master system.

NGLC : Total number of global cases of loading.

IDOF : Number of degrees of freedom per node = 3.

NODE : Number of nodes per element = 2.

MBEL : Total number of external boundary elements attached to the master system, if any.

UNIT : Conversion factor for length units, e.g. 12.0 if section and material properties are entered in 'inch' units, while the nodal geometry is entered in 'foot' units.

IDRY : If 1, this is a dry run.
If 0, this is a production run.

A.1.3 Substructure Orientation Cards (8F10.0)

If NSUB = 1 on card A.1.2, this group is to be omitted. This group of cards holds the orientation of the substructure local frames of reference relative to the master system reference frame. Enter as many cards as necessary to describe all substructures (8 substructure units/card).

10	20	30 … 80
		--- ORINT(I), I=1, NSUB ---

ORINT(I) : Local reference frame orientation relative to master system reference frame, measured in degrees (positive anti-clockwise).

A.1.4 Master System Connectivity Data Cards (20I4)

If NSUB = 1 on card A.1.2, this group is omitted. The group describes the connectivity of each substructure with the master system, and forms a table of width NSUB, and length (MXND + 1), where each column of the table holds the information for one substructure. Substructures must be ordered according to the global substructure identification number. An example of this table is shown in Table 4.1.

4	8	… 76	80
NPL(1)	NPL(2)	--- NPL(J) ---	
NPG(1,1)	NPG(1,2)	--- NPG(1,J) ---	
		--- NPG(K,J) ---	

J : The global number of the substructure.

NPL(J) : Number of inter-boundary nodes in this substructure.

NPG(K,J) : Node number in the master system to which node K of substructure J attaches, counting from the first inter-boundary node in this substructure.

NOTE: If there are more than 20 substructures, this table must be divided into a number of tables, entered successively, with the restriction that each table must contain (MXND + 1) cards.

A.1.5 Master System External Boundary Element Cards (3I4, F12.0)

One card per external boundary element. If MBEL = 0 on card A.1.2, this group is to be omitted. A boundary element is defined as a spring with very high stiffness attached to a node. Each 'element' can restrain one or more of the degrees of freedom at a node. Several elements with different stiffnesses in different directions can be attached to one node.

4	8	12	24	
N	MNPB(N)	MKODE(N)	AEB(N)	

N : Number of the boundary element.

MNPB(N) : Number of the master system node to which this element attaches.

MKODE(N) : A three digit number of the form ijk corresponding to degrees of freedom u, v, and r (where r is the nodal rotation), respectively. If any of these is 1, the corresponding degree(s) of freedom will be restrained; if any of these is 0, the corresponding degree(s) of freedom is (are) unrestrained.

AEB(N) : The spring stiffness of this element, if other than 10^{20}. The units should be consistent with the material properties.

A.2 Substructure Class Data Cards

This group will be repeated for every class of substructures. The types of data cards required for each class of substructure consists of the following:

A.2.1 Substructure Class Control Card

A.2.2 Nodal Geometry Cards

A.2.3 Member Data Cards

A.2.4 Substructure Load Case Identification Cards

A.2.5 External Boundary Element Cards

A.2.6 Joint Load/Displacement Cards

A.2.7 Member Loads

These cards are described in detail below.

A.2.1 Substructure Class Control Card (8I4)

4	8	12	16	20	24	28	32	
IC	NJ	NE	NLC	NSUBCL	NEBEL	NFIBN	NIBNS	

IC : Number of this class of substructure.

NJ : Total number of nodes for this class.

NE : Total number of elements (i.e. members) for this class.

NLC : Total number of independent cases of loading including zero cases of loading, for this class.

NSUBCL : Total number of substructures that belong to this class.

NEBEL : Total number of external boundary elements for this class.

NFIBN : Number of the first inter-boundary node for this class.

NIBNS : Total number of inter-boundary nodes for this class.

NOTE: NJ = NFIBN + NIBNS - 1

A.2.2 Nodal Geometry Cards (I4, 2F12.0, I4)

One card/node, unless automatic generation of nodal data is used.

4	16	28	32
N	X(N)	Y(N)	INC

N : node number.

X(N) : X-coordinate of the node.

Y(N) : Y-coordinate of the node.

INC : If non zero, automatic nodal generation will be initiated. The generated nodes will have numbers (NOLD + K*INC), where NOLD is the nodal number on the preceding card, and K is a positive integer. Generation terminates when (NOLD + K*INC) = N. The coordinates of the generated nodes are linearly interpolated between node NOLD and node N. INC should be positive.

NOTE: The first nodal geometry card must have INC = 0, and the last must have N = NJ.

A.2.3 Member Data Cards

A.2.3.1 Member Property Default Card (3F 12.0)

12	24	36	
ADEF	RDEF	YDEF	

ADEF : Default value for member area.

RDEF : Default value for moment of inertia of member.

YDEF : Default value for Young's modulus of member material.

Notice that if UNIT = 12.0 on card A.1.2, ADEF, RDEF, and YDEF should be entered in in.^{2}, in.^{4}, k/in.^{2} respectively. If UNIT = 1.0 any consistent set of units may be used.

A.2.3.2 Member Property and Connectivity Cards (7I4, 3F12.0)

One card per member, unless automatic member data generation is used.

4	8	12	16	20	24	28	40	52	64
M	NOD(I,M)	NOD(J,M)	MKODE(M)	INC	INCI	INCJ	AREA(M)	RI(M)	YMOD(M)

M : Member number. (NOTE: The last member card must have M = NE)

NOD(I,M) : End I node number.

NOD(J,M) : End J node number.

MKODE(M) : 0, if member is continuous at both ends.
1, if member is hinged at end I only.
2, if member is hinged at end J only.
3, if member is hinged at both ends.

INC : If non zero, automatic generation of member data will be initiated. The new members will have numbers (MOLD + K*INC), where MOLD is the member number on the preceding card, and K is a positive integer. These members will have the cross section and material properties of the member on the card initiating the generation (i.e. the present card). The end nodes of the new members will be (NOD(I,MOLD) + K*INCI) and (NOD(J,MOLD) + K*INCJ) for nodes at end I, J respectively. INC must be zero for the first member card of the substructure, and zero or positive otherwise.

INCI : Incrementation value for nodal number of end I.

INCJ : Incrementation value for nodal number of end J.

AREA(M) : Area of member section, if other than ADEF.

RI(M) : Moment of inertia of member section, if other than RDEF.

YMOD(M) : Young's modulus of member material, if other than YDEF.

NOTE: Units of AREA, RI, YMOD are the same as for ADEF, RDEF, and YDEF.

A.2.3.3 Echo Check Flag Card (I4)

4
K

K : If 1, the complete nodal geometry and member data will be contained in the output.
If 0, the output will not contain an echo check of the completed data.

A.2.4 Substructure Load Case Identification Cards (20I4)

This group identifies the global numbers of the substructures which belong to this class and the local cases of loading on each which form the global loading pattern for all global load cases. The group is composed of a table the first row of which holds the global identification numbers of the substructures starting in the second tabular column. The rest of the rows (cards) each hold the local load case numbers which define a global load case.

4	8	12	16 ... 76 80
0	NGSUB(1)	NGSUB(2)	--- (NGSUB(J), J=1, NSUBCL) ---
K	KSUB(K,1)	KSUB(K,2)	--- ((KSUB(K,J), J=1, NSUBCL), K=1, NGLC) ---

J : Subscript to span the total number of individual substructures of this class.

NGSUB(J) : Global identification number of this substructure.

K : Global load case number.

KSUB(K,J) : The local load case number that when imposed on global substructure number NGSUB(J), forms a part of global load case number K. These local substructure load cases will be specified in Sections. A.2.6 and A.2.7.

NOTE: If NSUBCL is greater than 19, this table must be divided into several tables, each composed of (NGLC + 1) cards.

A.2.5 External Boundary Elements Cards (3I4, F12.0)

If NEBEL = 0 on card A.2.1, this group is to be omitted. The group is composed of NEBEL cards, one card per boundary element. See Section A.1.5 for a definition of the external boundary elements.

4	8	12	24	
N	NPB(N)	KODE(N)	BES(N)	

N : Number of the external boundary element.

NPB(N) : Number of the substructure node to which this element is attached.

KODE(N) : A three digit number of the form ijk, corresponding to degrees of freedom u, v, and r, respectively. If any of these is 1, the corresponding degree of freedom will be restrained, otherwise the digit must be 0.

BES(N) : The spring stiffness of this element, if other than 10^{20}.

A.2.6 Joint Load/Displacement Cards

This group consists of a number of cards per loaded node, unless automatic load generation is used. The first card in a node group is a nodal identification card. The rest of the cards for this node each describe a local non-zero case of loading.

A.2.6.1 Nodal Identification Card (7I4)

4	8	12	16	20	24	78
N	NMLC	IDESP	NBEL(1)	NBEL(2)	NBEL(3)	INC

N : Node number (these need not be in order).

NMLC : Number of non-zero cases of loading at this node.

IDESP : If non-zero, displacements will be specified for one or more cases of loading.

(NBEL(I), I=1,3) : The boundary element numbers attached to this node, in case non-zero displacements are specified. These three numbers correspond to degrees of freedon u, v, and r, respectively. In case this node is restrained in 3 directions with one element, and non-zero displacements are specified only in 2 directions, then this boundary element number appears only twice in the corresponding locations.

INC : If non-zero, automatic loading generation will be initiated. Nodes with numbers (NOLD + K*INC), where NOLD is the node of the preceding nodal group, and K is a positive integer, will be assigned the same loads as on the node initiating the generation (N) for all local cases of loading. N must be greater than NOLD for successful load generation.

A.2.6.2 Nodal Load Cards (2I4, 3F12.0)

One card/non-zero case of loading. Number of cards of this group is NMLC, as input on card A.2.6.1.

4	8	20	32	44	
LC	JKODE(LC)	U(N,NLC)	V(N,NLC)	R(N,NLC)	

LC : Number of local load case (need not be in order).

JKODE(LC) : A three digit number of the form ijk, the digits corresponding respectively to u, v, and r. If any digit is 1, the corresponding u, v or r will be interpreted as a displacement in the direction of the corresponding NBEL of card A.2.6.1, otherwise it must be zero.

U(N,NLC) : Load in the direction of the X-axis.

V(N,NLC) : Load in the direction of the Y-axis.

R(N,NLC) : Moment in the X-Y plane, positive anti-clockwise.

NOTE: If UNIT = 12, and YMOD is in K/in.^{2}, u, and v must be in kips and r in K.ft. Also notice that the u, v, and r arrays do not exist in the program. The corresponding values are entered directly in the local load array, B.

A.2.6.3 Termination Card

One blank card at the end of the 'joint load/displacement card group'.

A.2.7 Member Load Specification Cards

This group consists of a number of cards for each element on which non zero loads are applied, unless automatic load generation is

used. Any number of loads can be applied on an element in one or more load cases. Load generation is carried out for one load for one case of loading at a time, and it can be of two types. The first type generates loads on elements which do not have the same end conditions, nor the same orientation or length. The second type involves members which have the same orientation and length, but not necessarily the same end conditions. In cases other than when automatic load generation is used, the cards may be entered in any order, since any one card defines completely the member number, the local load case number, the load, and its type, position, and orientation with respect to the member.

A.2.7.1 Member Load Cards (4I4, 2F12.0, 4F10.0)

One card per member per load per load case, unless automatic load generation is used.

4	8	12	16	28	40	50	60	70	80
M	LC	INC	K	CL	CN	QI	AI	QJ	AJ

M : Member number.

LC : Local load case number to which this load belongs.

INC : If non zero, automatic load generation will be initiated for load case LC. The load on the card initiating the generation, or its end effects, will be assigned to members with numbers (MOLD + L*DABS(INC)), where MOLD is the member number on the preceding card, which must have the same LC value, and L is a positive integer.

If INC is positive, the generation is of the first type, and if INC is negative, the generation will be of the second type described in Section A.2.7.

K : If 1, the load is concentrated

If 2, the load is of uniform intensity

If 3, the load is trapezoidal.

CL : Projection along the member of a vector in direction of load.

CN : Projection normal to the member of a vector in direction of load.

QI : Magnitude of load at distance (AI*length) from end I.

AI : Fraction of length from end I to beginning of load.

QJ : Magnitude of load at distance (AJ*length) from end I.

AJ : Fraction of length from end I to end of load.

NOTE: If the load is concentrated, QJ and AJ may be omitted. If the load is uniform, QJ may be omitted. Units of load are in K, or K/ft., if UNIT = 12.0.

A.2.7.2 Termination Card

One blank card at the end of the group of member load cards. (The program returns to A.2.1 for input data for the next substructure).

A.3 Common Block Description

The size of the problem is passed to the data manager through the MAIN segment, as indicated in Section 4.2.1. The calculation of the sizes of the different common blocks should be carried out prior to run time and values assigned to array ICOM in segment MAIN according to the following formulas. The values of NWK and NWA may be set to zero for the dry run described in Section A.3.6. The dry run outputs the values of these variables to allow recomputation of common block sizes prior to the production run.

A.3.1 Size of Common Block MASTRA

ICOM(1) = NSUB + IDOF*NUMNST*NGLC + MBEL + NWK

A.3.2 Size of Common Block MASTIA

ICOM(2) = NAUB*(MXND + 1) + IDOF*(2*NUMNST + MXND)
+ 2*(MBEL + NEBEL) + 1

A.3.3 Size of Common Block SUBRA

ICOM(3) = IDOF*NLC*(NJ + NODE*NE) + 3*NE + 2*NJ + NWA

A.3.4 Size of Common Block SUBIA

ICOM(4) = NE*(NODE + 1) + NSUBCL*(NGLC + 1) + 2*IDOF*NJ
+ IDOF*NODE + 1

A.3.5 Size of Common Block SUBSR

ICOM(5) = IDOF*(NODE*NE + NGLC*NJ)

A.3.6 The Dry Run Facility

This option is implemented to check the input data, and to calculate the exact sizes of the stiffness storage vectors NWK and NWA. The option is exercised by specifying IDRY as indicated in Section A.1.2. During this run the sizes of the common blocks MASTRA, and SUBRA are input with zero values of NWK and NWA, where NWK is the size of the master system stiffness vector, and NWA is the size of a class of substructure stiffness vector. The dry run will give these values, which are then used to update the corresponding common block sizes prior to the production run. For this purpose the upper bound of ICOM(3), ICOM(4) and ICOM(5) values over all classes of substructure should be used.

APPENDIX B

USER'S MANUAL FOR MUSAPF

APPENDIX B - USER'S MANUAL FOR MUSAPF

B.1 Problem Control Cards

B.1.1 Heading Card (20A4)

One card which contains any title for the problem.

B.1.2 Problem Control Parameter Card (8I4, F12.0, I4)

4	8	12	16	20	24	28	32	44	48
NSUB	NINDSB	NTOTAS	NINDAS	NGLC	NMBKSB	IDOF	NODE	UNIT	IDRY

NSUB : Total number of basic substructure units.

NINDSB : Total number of independent basic units

NTOTAS : Total number of substructures and substructure assemblies including the master system.

NINDAS : Total number of independent units including the basic independent units, the assembled units, and the master system.

NGLC : Total number of global cases of loading.

NMBKSB : Number of backsubstitution steps.

IDOF : Number of degrees of freedom per node = 3.

NODE : Number of nodes per element = 2.

UNIT : Conversion factor for length, e.g. 12.0 if the section and material properties are in 'inch' units, while nodal geometry is in 'feet' units.

IDRY : If 1, this is a dry run.
If 0, this is a production run.

B.2 Backsubstitution Control Cards (20I4)

This group of cards describes the backsubstitution and is similar to Table 4.6. The table is composed of four rows (cards) and NMBKSB columns. If NMBKSB > 20 the table must be divided into several tables each consisting of four cards. If NTOTAS = 1, the problem is unsubstructured and this group must be omitted.

4	8	4*I	76	80
		--- INF1BK(I) ---		
		--- INF2BK(I) ---		
		--- INF3BK(I) ---		
		--- INF4BK(I) ---		

I : Subscript denoting the number of the backsubstitution step.

INF1BK(I) : Global number of a higher level unit for which a solution vector has been determined, e.g. INF1BK(1) = NTOTAS.

INF2BK(I) : Global number of a lower level unit for which a solution vector is sought. It must be < NTOTAS.

INF3BK(I) : Number of the independent unit which represents the lower level unit. It must be < NINDAS.

INF4BK(I) : Local number of the lower level unit in the higher level unit control arrays.

B.3 Independent Basic Unit Data Cards

This group will be repeated for every independent basic unit. The types of data cards required for each unit are as follows:

- B.3.1 Independent Basic Unit Control Card
- B.3.2 Nodal Geometry Cards
- B.3.3 Member Data Cards
- B.3.4 External Boundary Element Cards
- B.3.5 Loading Data Cards

These cards are described in detail below.

B.3.1 Independent Basic Unit Control Card (7I4)

4	8	12	16	20	24	28
IC	NJ	NE	NLC	NEBEL	NFIBN	NIBNS

IC : Independent basic unit number. The units must be entered in order.

NJ : Total number of nodes for this unit.

NE : Total number of elements for this unit.

NLC : Total number of independent cases of loading for this unit. If a zero loading on the unit occurs in the global loading pattern, the zero loading must be considered a case of loading. Thus, if this unit is not loaded at all NLC = 1.

NEBEL : Total number of external boundary elements, if any.

NFIBN : Number of first inter-boundary node.

NIBNS : Total number of inter-boundary nodes for this unit.

B.3.2 Nodal Geometry Cards (I4, 2F12.0, I4)

One card per node unless automatic nodal generation is used.

4	16	28	32
N	X(N)	Y(N)	INC

The definition of these variables can be found in Section A.2.2.

B.3.3 Member Data Cards

B.3.3.1 Member Property Default Card (3F12.0)

12	24	36	
ADEF	RDEF	YDEF	

The definition of these symbols can be found in Section A.2.3.1.

B.3.3.2 Member Property and Connectivity Cards (7I4, 3F12.0)

One card per member unless automatic member data generation is used

4	8	12	16	20	24	28	40	52	64
M	NOD(I,M)	NOD(J,M)	MKODE(M)	INC	NODI	NODJ	AREA(M)	RI(M)	YMOD(M)

The definition of these variables can be found in Section A.2.3.2.

B.3.3.3 Echo Check Flag Card (I4)

4
K

K : If 1, the complete nodal geometry and member data is output. If 0, the output will not contain an echo check of the completed data.

B.3.4 External Boundary Element Cards (3I4, F12.0)

One card per external boundary element. If NEBEL = 0 on card

B.3.1, this group is to be omitted. The definition of a boundary element can be found in Section A.1.5.

4	8	12	24
N	NPB(N)	KODE(N)	BES(N)

The definition of these variables can be found in Section A.2.5.

B.3.5 Load Data Cards

B.3.5.1 Load Flag Card (2I4)

4	8
JLFLAG	MLFLAG

JLFLAG : If zero, no joint loads will be prescribed on this unit.
If non-zero, joint loads will be prescribed on this unit.

MLFLAG : If zero, no member loads will be prescribed on this unit.
If non-zero, member loads will be prescribed on this unit.

B.3.5.2 Joint Load/Displacement Cards

This group consists of a number of cards per loaded node, unless automatic load generation is used. The first card in a nodal load group is a node identification card. The rest of the cards in a nodal group, each describe a local non-zero case of loading. If JLFLAG = 0 on card B.3.5.1, this group is omitted.

B.3.5.2.1 Node Identification Card (7I4)

One card at the head of a nodal load group.

4	8	12	16	20	24	28	
N	NMLC	IDESP	NBEL(1)	NBEL(2)	NBEL(3)	INC	

The definition of these variables can be found in Section A.2.6.1.

B.3.5.2.2 Nodal Load Cards (2I4, 3F12.0)

One card per non-zero case of loading on the node identified on card B.3.5.2.1.

4	8	20	32	44
LC	JKODE(LC)	U(N,NLC)	V(N,NLC)	R(N,NLC)

The definition of these variables can be found in Section A.2.6.2.

B.3.5.2.3 Termination Card

One blank at the end of the joint load/displacement card group.

B.3.5.3 Member Load Specification

This group consists of a number of cards for each member on which non-zero loads are applied, unless automatic load generation is used. Any number of loads can be applied on an element in one or more load cases. Load generation is carried out for one load for one local case of loading at a time, and it can be of two types. The first type generates loads on members which do not have the same end conditions, nor the same orientation or length. The second type involves members which have the same orientation and length, but not necessarily the same end conditions. In cases other than when automatic load generation is

used, the cards may be entered in any order, since any one card defines completely the member number, the local load case number, the load, and its type, position, and orientation with respect to the member. If MLFLAG = 0 on card B.3.5.1, this group is omitted.

B.3.5.3.1 Member Load Cards (4I4, 2F12.0, 4F10.0)

One card per member per load per load case, unless automatic load generation is used.

4	8	12	16	28	40	50	60	70	80
M	LC	INC	K	CL	CN	QI	AI	QJ	AJ

The definition of these variables can be found in Section A.2.7.1.

B.3.5.3.2 Termination Card

One blank card at the end of the group of member load cards. (The program returns to B.3.1 for input data for the next independent basic unit, unless IC on card B.3.1 for this group = NINDSB, in which case it proceeds to B.4 for the independent assembled unit data cards).

B.4 Independent Assembled Unit Data Cards

This group must be repeated for each independent assembled unit. They must be in order, ending with the master system. If the problem is unsubstructured (NSUB = NTOTAS = 1), this group is to be omitted. The types of cards required for each independent assembled unit consists of the following:

B.4.1 Independent Assembled Unit Control Card

B.4.2 Constituent Unit Information Cards

B.4.3 External Boundary Element Cards.

These cards are described in detail below.

B.4.1 Independent Assembled Unit Control Card (9I4)

4	8	12	16	20	24	28	32	36	
ICH	NJH	NLCH	NEBELH	NFIBNH	NIBNSH	NCU	MXND	IFLAG	

ICH : Independent assembled unit number. These are a continuation of the IC numbers on card B.3.1, and are entered in order ending with the master system which must have ICH = NINDAS. If any of the constituent units is itself an assembled unit, it must have a lower ICH number, i.e. it must have already been read and defined.

NJH : Total number of nodes for this unit.

NLCH : Total number of independent load cases for this unit.

NEBELH : Total number of external boundary elements for this unit.

NFIBNH : Number of the first inter-boundary node for this unit.

NIBNSH : Total number of inter-boundary nodes in this unit.

NCU : Total number of constituent units for this assembled unit.

MXND : Maximum number of inter-boundary nodes in any of the constituent

units for this assembled unit.

IFLAG : This variable is only meaningful for the IAU which constitutes the master system. It should be non-zero only if the master system solution vector is to be stored for a backsubstitution that requires this vector and occurs after backsubstitution for lower level units has commenced.

B.4.2 Constituent Unit Information Cards

B.4.2.1 Connectivity and Load Case Identification Cards (2014)

This group forms a table of width NCU and length (MXND + 3 + NLCH), where these variables are defined on card B.4.1, where every column holds the necessary connectivity information for one constituent unit. If the number of constituent units NCU is greater than 20, the table must be split into several tables with the restriction that each table must have (MXND + 3 + NLCH) cards.

4	8 4*I 76	80
INDSUB(1)	--- INDSUB(I), I=1, NCU ---	
IRCSUB(1)	--- IRCSUB(I), I=1, NCU ---	
IBNSUB(1)	--- IRNSUB(I), I=1, NCU ---	
NPG(1,1)	--- NPG(1,I), I=1, NCU ---	
NPG(2,1)	--- NPG(2,I), I=1, NCU ---	
	((NPG(J,I), I=1, NCU), J=1, MXND)	
LCSUB(1,1)	--- LCSUB(1,I), I=1, NCU ---	
LCSUB(K,1)	--- ((LCSUB(K,I), I=1, NCU), K=1, NLCH) ---	

I : A subscript which denotes the position or local number of a constituent unit in the assembled unit ICH.

INDSUB(I) : Identification number of the independent unit which represents the constituent unit I. INDSUB(I) < ICH.

IRCSUB(I) : If zero, it means that the arrays of independent unit INDSUB(I) are already in core. This flag helps to eliminate the time needed to retrieve these arrays from backing storage, in case this independent unit represents more than one constituent unit. Such constituent units should occupy adjacent columns of this table with the first having IRCSUB(I) = 1.

IBNSUB(I) : Total number of inter-boundary nodes for constituent unit INDSUB(I).

NPG(J,I) : Number of the assembled unit ICH node to which node J of constituent unit INDSUB(I) attaches, counting from the first inter-boundary node in the lower level system.

K : A subscript which denotes the load case for the assembled unit ICH.

LCSUB(K,I): The identification number of the independent load case out of system INDSUB(I), which when applied on constituent unit I forms a part of the loading scheme for load case K on assembled unit ICH.

B.4.2.2 Constituent Unit Orientation Cards (8F10.0)

Any number of cards to describe the orientation of the local frames of reference of the constituent units. (Eight units per card).

10	10*I	80
ORINT(1)	--- ORINT(I) , I = 2, NCU	

I : A subscript which denotes the position or local number of a constituent unit in the assembled unit ICH control arrays.

ORINT(I) : The orientation of the frame of reference of constituent unit I relative to the frame of reference of the assembled unit ICH, measured in degrees (positive counter-clockwise).

B.4.3 External Boundary Element Cards (3I4, F12.0)

One card per external boundary element. If NEBELH = 0 on card B.4.1, this group is to be omitted. The definition of an external boundary element can be found in Section A.1.5.

4	8	12	24	
N	NPB(N)	KODE(N)	BES(N)	

The definition of these variables can be found in Section A.2.5. (The program now returns to B.4.1 for the input data of the next assembled unit, unless ICH = NINDAS, i.e. the last unit).

B.5 Common Block Description

The size of the problem is passed to the data manager through the MAIN segment, as indicated in Section 4.3.1. The calculation of the sizes of the different common blocks should be carried out prior to run time, and values assigned to array ICOM in segment MAIN according to the following formulas. Where more than one formula is given, the largest requirement governs. The values of NWK, and NWA may be set to zero for the dry run described in Section B.5.6. The dry run outputs the values of these variables to allow the recomputation of common block sizes prior to the production run.

B.5.1 Size of Common Block PROBIA

ICOM(1) = 4*NMBKSB + 3*NTOTAS + 1

B.5.2 Size of Common Block RA1

ICOM(2) = 2*NJ + 3*NE + IDOF*NODE*NLC*NE + NEBEL

B.5.3 Size of Common Block IA1

ICOM(3) = NE*(NODE + 1) + 2*NEBEL + NLC

or ICOM(3) = NCU*(MXND + NLCH + 3) + 2*NEBELH

B.5.4 Size of Common Block RA2

ICOM(4) = NCU + NJH*IDOF*NLCH + NWK + NJ*IDOF*NLC + NWA

or ICOM(4) = NCU + NJH*IDOF*NGLC + NWA + NJ*IDOF*(NLC + NGLC) + NE*NODE*IDOF.

B.5.5 Size of Common Block IA2

ICOM(5) = IDOF*(NOD + 1 + 2*NJH)

or ICOM(5) = IDOF*(2*NJH + NJ + MXND)

or ICOM(5) = IDOF*NJ + NE*(NODE + 1)

or ICOM(5) = NCU*(MXND + NLC + NGLC)

B.5.6 The Dry Run Facility

This option is implemented to check the input data, and to calculate the exact sizes of the stiffness storage vectors NWK and NWA. The option is exercized by specifying IDRY = 1 as indicated in Section B.1.2. During this run the size of the common block RA2 is calculated with zero values for NWK and NWA, where NWK is the size of the stiffness storage vector for an assembled unit, and NWA is the size of the stiffness storage vector for a lower level unit. The dry run will output these values, which are then used to update the common block RA2 size prior to the production run. For these values the upper bounds for all the common block sizes over all possible configurations of the involved variables should be used.

APPENDIX C

PROGRAM DESCRIPTIONS AND LISTINGS

APPENDIX C - PROGRAM DESCRIPTIONS AND LISTINGS

C.1 Program SISAPF

Program SISAPF consists of the following parts.

1. MAIN
2. MAINMG, the main executive subroutine.
3. The input package

   INPUT1, INPUT2, INPUT3, and INPUT4
4. The data managing package

   ISPAC, LOCOM, REMOV, REMOV2, and BLOCK DATA.
5. The data storage and retrieval package

   CLEAR, ICLEAR, RTRV1, RTRV2, STORE1, and STORE2.
6. The equation solving package

   ADDRES, COLHT, EQSBST, EQFT, EQKBB, BKSB1, and BKSB2.
7. The formulation and output package

   ASSEMB, BOUND, BOUND2, DISPL, JLOAD, MLOADS, RESUB, STIFF, and STRESS.

The listing for the first three packages follows. The listing for packages 4 to 7 can be found in Sections C.3, C.4, C.5, and C.6 respectively.

```
C234567890123456789012345678901234567890123456789012345678901234
C
C                               SISAPP
C
C   THIS IS A PROGRAM FOR SINGLE LEVEL SUBSTRUCTURE ANALYSIS OF
C   PLANER FRAMED TYPE STRUCTURES. NO LIMITATIONS ARE PLACED ON
C   THE NUMBER OF, OR THE NUMBER OF CLASSES OF SUBSTRUCTURES,
C   OR THE NUMBER OF CASES OF LOADING. THE EQUATION SOLVER IS
C   OF SKYLINE IN-CORE TYPE SOLVER.
C
C*****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 NAMES
C
      COMMON /MASTIA/ NNN(1000)
      COMMON /MASTRA/ BBB(15000)
      COMMON /SUBIA/  MMM(4000)
      COMMON /SUBRA/  AAA(34000)
      COMMON /SUBSR/  CCC(4000)
      COMMON /DIMCOM/ L1,L2,L3,L4,L5,MXDM,NAMES(5,20),IPT(5,21),
     *ICOM(5)
C
      ICOM(1) = 15000
      ICOM(2) = 1000
      ICOM(3) = 34000
      ICOM(4) = 4000
      ICOM(5) = 4000
C
      CALL MAINMG
C
      END
```

```
      SUBROUTINE MAINMG
C
C   THIS SUBROUTINE MANAGES THE READING,THE FORMULATION,AND THE
C   SOLUTION OF THE PROBLEM. PROGRAM SISAPP.
C
C*****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 NAMES,NAME
C
      COMMON /MASTIV/ UNIT,NSUB,NCLSUB,MXND,NUMNST,NGLC,IDOF,
     *NODE,MBEL,IDRY,IN,IO
      COMMON /MASTIA/ NNN(1)
      COMMON /MASTRA/ BBB(1)
      COMMON /SUBIV/ NJ,NE,NLC,NSUBCL,NEBEL,NFIBN,NIBNS,ICLSUB
      COMMON /SUBIA/ MMM(1)
      COMMON /SUBRA/ AAA(1)
      COMMON /SUBSR/ CCC(1)
      COMMON /DIMCOM/ LA1,LA2,LA3,LA4,LA5,MXDIM,NAMES(5,20),
     *IPT(5,21),ICOM(5)
C
C*****************************************************************
      IN = 5
      IO = 6
      REWIND 1
      CALL TIME(0,0)
C
C   READ MASTER STRUCTURE CONTROL VARIABLES.
      CALL INPUT1
      IF(NSUB.EQ.1) GO TO 20
C
C   READ MASTER STRUCTURE CONNECTIVITY AND SUBSTRUCTURE
C   ORIENTATIONS.
      I1 = ISPAC(5HORINT,NSUB,1)
      J1 = ISPAC(3HNPG,(MXND*NSUB),2)
      J2 = ISPAC(3HNPL,NSUB,2)
      CALL INPUT2(BBB(I1),NNN(J1),NNN(J2))
C
C   READ MASTER STRUCTURE EXTERNAL BOUNDARY CONDITIONS.
      IF(MBEL.EQ.0) GO TO 5
      J3 = ISPAC(4HMNPB,MBEL,2)
      J4 = ISPAC(6HMSKODE,MBEL,2)
      I4 = ISPAC(3HAEB,MBEL,1)
      CALL BOUND(BBB(I4),NNN(J3),NNN(J4),MBEL,IN,IO)
 5    WRITE(IO,2000)
      CALL TIME(3,3)
C
C   FORM COLUMN HEIGHT AND ADDRESSING ARRAY FOR MASTER STRUCTURE
      J5 = ISPAC(4HMAXB,(NUMNST*IDOF+1),2)
      J6 = ISPAC(3HMHB,(NUMNST*IDOF),2)
      CALL ICLEAR(NNN(J6),(NUMNST*IDOF))
C
      DO 10 I=1,NSUB
      NODES = NNN(J2+I-1)
      ND    = NODES*IDOF
      J7 = ISPAC(2HLM,ND,2)
```

```
      CALL COLHT(MXND,NODES,ND,IDOF,I,NNN(J6),NNN(J1),NNN(J7))
      CALL REMOV(2HLM,2)
10    CONTINUE
C
      CALL ADDRES(NNN(J5),NNN(J6),NUMNST,IDOF,NWK)
      IF(IDRY.EQ.1) GO TO 15
C
C   RESERVE SPACE FOR MASTER ARRAYS BP,...
      I2 = ISPAC(2HBP,(IDOF*NUMNST*NGLC),1)
      I3 = ISPAC(2HBB,NWK,1)
      CALL  CLEAR(BEE(I2),(NUMNST*IDOF*NGLC+NWK))
C
15    WRITE(IO,2100) NWK
      CALL TIME(3,3)
C
C   LOOP OVER SUBSTRUCTURE CLASSES TO READ DATA, FORM AND PARTI-
C   ALLY REDUCE STIFFNESS MATRIX AND LOAD VECTOR, AND ASSEMBLE
C   THE CORRESPONDING SUBSTRUCTURE PARTITIONS INTO THE MASTER
C   ARRAYS, WITH THE APPROPPIATE TRANSFORMATIONS.
C
20     ICLSUB = 1
C
C   READ SUBSTRUCTURE CLASS(ICLSUB) CONTROL DATA.
30     CALL INPUT3
C
C   READ SUBSTRUCTURE CLASS(ICLSUB) GEOMETRY,MEMBER CONNECTIVITY
C   AND PROPERTIES.
       K1 = ISPAC(1HX,NJ,3)
       K2 = ISPAC(1HY,NJ,3)
       K3 = ISPAC(4HAREA,NE,3)
       K4 = ISPAC(2HRI,NE,3)
       K5 = ISPAC(4HYMOD,NE,3)
       L1 = ISPAC(3HNOD,(NODE*NE),4)
       L2 = ISPAC(5HMKODE,NE,4)
       L3 = ISPAC(4HKSUB,(NSUBCL*(NGLC+1)),4)
       L4 = ISPAC(5HNGSUB,NSUBCL,4)
       CALL INPUT4(AAA(K1),AAA(K2),AAA(K3),AAA(K4),AAA(K5),
      *MMM(L1),MMM(L2),MMM(L3),MMM(L4))
C
C   READ EXTERNAL BOUNDARY CONDITIONS IF ANY.
       IF(NEBEL.EQ.0) GO TO 40
       J8 = ISPAC(3HNEB,NEBEL,2)
       J9 = ISPAC(4HKODE,NEBEL,2)
       M1 = ISPAC(2HEE,NEBEL,3)
       GO TO 50
40     M1 = 1
       GO TO 60
50     CALL BOUND(CCC(M1),NNN(J8),NNN(J9),NEBEL,IN,IO)
C
C   READ JOINT LOADS.
60     L5 = ISPAC(5HJKODE,NLC,4)
       K6 = ISPAC(3HFEF,(IDOF*NODE*NE*NLC),3)
       K7 = ISPAC(1HB,(IDOF*NJ*NLC),3)
       CALL CLEAR(AAA(K6),(IDOF*NLC*(NJ+NE*NODE)))
       CALL JLOAD(MMM(L5),AAA(K7),CCC(M1),NLC,IN,IO)
       CALL REMOV(5HJKODE,4)
C
C   READ AND PREPARE MEMBER LOADS.
       CALL MLOADS(AAA(K1),AAA(K2),AAA(K6),AAA(K7),MMM(L1)
      *,MMM(L2),IDRY,UNIT,IN,IO,NLC,NJ)
C
       WRITE(IO,2200) ICLSUB
       CALL TIME(3,3)
C
C   FORM THE COLUMN HEIGHTS AND ADDRESSING ARRAYS .
       L6 = ISPAC(4HMAXA,(IDOF*NJ+1),4)
       L7 = ISPAC(3HMHT,(IDOF*NJ),4)
       L8 = ISPAC(2HLM,(IDOF*NODE),4)
       CALL ICLEAR(MMM(L7),(IDOF*NJ))
C
       DO 70 I=1,NE
       CALL COLHT(NODE,NODE,(NODE*IDOF),IDOF,I,MMM(L7),MMM(L1),
      *MMM(L8))
70     CONTINUE
       CALL REMOV(2HLM,4)
       CALL ADDRES(MMM(L6),MMM(L7),NJ,IDOF,NWA)
C
       WRITE(IO,2300) ICLSUB,NWA
       CALL TIME(3,3)
       IF(IDRY.EQ.1) GO TO 95
       CALL REMOV(3HMHT,4)
C
C   RESERVE SPACE FOR STIFFNESS MATRIX.
       K8 = ISPAC(1HA,NWA,3)
       CALL CLEAR(AAA(K8),NWA)
C
C   FORM AND ASSEMBLE MEMBER STIFFNESSES INTO A.
       CALL STIFF(MMM(L1),MMM(L2),MMM(L6),AAA(K1),AAA(K2),AAA(K3)
      *,AAA(K4),AAA(K5),AAA(K8),UNIT,NE)
C
C   ADD EXTERNAL BOUNDARY CONDITIONS IF ANY TO,  TO MATRIX A.
       IF(NEBEL.EQ.0) GO TO 80
       CALL BOUND2(AAA(K8),MMM(L6),NNN(J8),NNN(J9),CCC(M1),NEBEL)
C
80     WRITE(IO,2400) ICLSUB
       CALL TIME(3,3)
C
C   REDUCE A NAD B TO LEVEL OF PARTITION IF ANY.
C
       NI = NJ - NIBNS
       IF(NI.EQ.0) GO TO 85
       CALL EQSBST(AAA(K8),AAA(K7),MMM(L6),NI,IDOF,NLC)
       IF(NSUB.EQ.1) GO TO 100
C
       CALL EQPT(AAA(K8),MMM(L6),NI,IDOF,NJ)
C
       CALL EQKBB(AAA(K8),AAA(K7),MMM(L6),NI,IDOF,NJ,NLC)
       CALL TIME(3,3)
C
C   ASSEMBLE STIFFNESS AND LOAD VECTOR INTER-BOUNDARY PARTITIONS
```

```
C   INTO THE MASTER ARRAYS WITH THE APPROPRIATE TRANSFORMATIONS.
C
85     DO 90 I=1,NSUBCL
       J = MMM(L4-1+I)
       CALL ASSEMB(NNN(J1),NNN(J5),MMM(L3),MMM(L6),BBB(I1),BBB(I2
      *),BBB(I3),AAA(K7),AAA(K8),J,I,NFIBN,NIBNS,MXND,NLC,NGLC)
90     CONTINUE
C
       WRITE(IO,2500) ICLSUB
C
       IF(NCLSUB.EQ.1) GO TO 97
C
C WRITE POINTERS AND SUBSTRUCTURE CLASS COMMON BLOCKS ON FILE(1)
       II1 = IPT(4,LA4+1) - 1
       II2 = IPT(3,LA3+1) - 1
       WRITE(1) K1,K2,K3,K4,K5,K6,K7,K8,L1,L2,L3,L4,L6
       WRITE(1) NJ,NE,NLC,NSUBCL,NFIBN,NIBNS,ICLSUB,II1,II2
       CALL STORE1(1,MMM(1),II1)
       CALL STORE2(1,AAA(1),II2)
95     CALL REMOV2(3)
       CALL REMOV2(4)
       IF(NEBEL.EQ.0) GO TO 97
       CALL REMOV(4HKODE,2)
       CALL REMOV(3HNPB,2)
       CALL REMOV(2HEB,5)
C
97     WRITE(IO,2600) ICLSUB
       CALL TIME(3,3)
       IF(ICLSUB.EQ.NCLSUB) GO TO 120
       ICLSUB = ICLSUB + 1
       GO TO 30
C
C*****************************************************************
C
C   THIS IS A BRANCH TO AN UNSUBSTRUCTURED PROBLEM. PROCEED WITH
C   BACKSUBSTITUTION FROM THE TRIANGULARIZED STIFFNESS MATRIX,
C   AND MEMBER END FORCES CALCULATIONS.
C
100    CALL BKSB1(AAA(K7),AAA(K8),MMM(L6),NI,NLC,IDOF,0)
C
       M2 = ISPAC(4HFEF2,IDOF*NODE*NE,5)
C
       WRITE(IO,2700)
       CALL TIME(3,3)
C
       DO 110 I=1,NLC
       CALL DISPL(CCC(M2),AAA(K7),AAA(K6),MMM(L3),I,1,1,NJ,NE,NLC
      *,NLC,IN,IO,NSUB)
C
       CALL STRESS(AAA(K1),AAA(K2),AAA(K3),AAA(K4),AAA(K5),
      *AAA(K7),CCC(M2),MMM(L1),MMM(L2),NE,I,NLC,UNIT,IN,IO)
110    CONTINUE
C
       WRITE(IO,2800)
       CALL TIME(3,3)
C
       RETURN
C
C*****************************************************************
C
C   ADD EXTERNAL BOUNDARY CONDITIONS TO MASTER STIFFNESS MATRIX.
120    IF(IDRY.EQ.1) GO TO 170
       IF(MBEL.EQ.0) GO TO 125
       CALL BOUND2(BBB(I3),NNN(J5),NNN(J3),NNN(J4),BEB(I4),MBEL)
C
C   SOLVE MASTER SYSTEM OF A SUBSTRUCTURED PROBLEM.
125    CALL EQSBST(BBB(I3),BBB(I2),NNN(J5),NUMNST,IDOF,NGLC)
C
       CALL BKSB1(BBB(I2),BBB(I3),NNN(J5),NUMNST,NGLC,IDOF,0)
C
       WRITE(IO,2900)
       CALL TIME(3,3)
C
C*****************************************************************
C
C   LOOP OVER SUBSTRUCTURE CLASSES TO COMPUTE AND OUTPUT
C   SUBSTRUCTURE DISPLACEMENTS, AND MEMBER END FORCES.
C
       IF(NCLSUB.EQ.1) GO TO 140
       REWIND 1
C
130    READ(1) K1,K2,K3,K4,K5,K6,K7,K8,L1,L2,L3,L4,L6
       READ(1) NJ,NE,NLC,NSUBCL,NFIBN,NIBNS,ICLSUB,II1,II2
       CALL RTRV1(1,MMM(1),II1)
       CALL RTRV2(1,AAA(1),II2)
C
140    M2 = ISPAC(4HFEF2,IDOF*NODE*NE,5)
       M3 = ISPAC(2HBA,IDOF*NJ*NGLC,5)
C
       DO 160 J=1,NSUBCL
       I = MMM(L4-1+J)
       CALL RESUB(AAA(K7),CCC(M3),BBB(I1),BBB(I2),NNN(J1),MMM(L3)
      *,I,J,NGLC,MXND,NFIBN,NIBNS,NLC)
       NI = NJ - NIBNS
       IF(NI.EQ.0) GO TO 145
       CALL BKSB2(AAA(K8),CCC(M3),MMM(L6),NFIBN,NJ,NGLC,IDOF)
       CALL BKSB1(CCC(M3),AAA(K8),MMM(L6),NI,NGLC,IDOF,1)
C
145    DO 150 K=1,NGLC
       CALL DISPL(CCC(M2),CCC(M3),AAA(K6),MMM(L3),K,J,I,NJ,NE,NLC
      *,NGLC,IN,IC,NSUB)
C
       CALL STRESS(AAA(K1),AAA(K2),AAA(K3),AAA(K4),AAA(K5),
      *CCC(M3),CCC(M2),MMM(L1),MMM(L2),NE,K,NGLC,UNIT,IN,IC)
150    CONTINUE
C
160    CONTINUE
C
       CALL REMOV2(5)
C
```

```
      WRITE(IO,3200) ICLSUB
      CALL TIME(3,3)
      ICLSUB = ICLSUB + 1
      IF(ICLSUB.LE.NCLSUB) GO TO 130
C
170   RETURN
C
C   FORMAT STATEMENTS
C
2000  FORMAT(//'MASTER STRUCTURE DATA INPUT IS COMPLETED')
2100  FORMAT(//'MASTER ADDRESSING ARRAY IS FORMED. NWK=',I4)
2200  FORMAT(//' SUBSTRUCTURE CALSS(',I4,') INPUT IS COMPLETED')
2300  FORMAT(//'SUBSTRUCTURE CALSS(',I4,') ADDRESSING ARRAY IS',
     *' COMPLETED.      NWA=',I4)
2400  FORMAT(//'SUBSTRUCTURE CLASS(',I4,') STIFFNESS MATRIX IS',
     *' COMPLETED.')
2500  FORMAT(//'SUBSTRUCTURE CLASS(',I4,') STIFFNESS AND LOAD',
     *' VECTOR REDUCTION AND ASSEMBLY IS COMPLETED.')
2600  FORMAT(//'SUBSTRUCTURE CLASS(',I4,') DATA IS STORED ON',
     *' FILE(1).')
2700  FORMAT(//'DISPLACEMENTS COMPUTATIONS ARE COMPLETED')
2800  FORMAT(//'STRESS COMPUTATIONS ARE COMPLETED')
2900  FORMAT(//'MASTER STRUCTURE SOLUTION IS COMPLETED')
3200  FORMAT(//'COMPUTATIONS FOR SUBSTRUCTURE CLASS(',I4,
     *') ARE COMPLETEED')
C
       END
```

```
      SUBROUTINE INPUT1
C
C   THIS SUBROUTINE READS CONTROL DATA OF MASTER STRUCTURE.
C
C***********************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      COMMON /MASTIV/ UNIT,NSUB,NCLSUB,MXND,NUMNST,NGLC,IDOF,
     *NODE,MBEL,IDRY,IN,IO
C
      DIMENSION HED(20)
C
      READ(IN,1000) HED
      WRITE(IO,2000) HED
C
      READ(IN,1100) NSUB,NCLSUB,MXND,NUMNST,NGLC,IDOF,NODE,
     *MBEL,UNIT,IDRY
      WRITE(IO,2100) NSUB,NCLSUB,MXND,NUMNST,NGLC,IDOF,NODE,
     *MBEL,UNIT,IDRY
C
      RETURN
C
C   FORMAT STATEMENTS
C
1000  FORMAT(20A4)
1100  FORMAT(8I4,F12.0,I4)
2000  FORMAT(20X,20A4///)
2100  FORMAT('MASTER STRUCTURE CONTROL VARIABLES'//,
     *'NUMBER OF SUBSTRUCTURES                          =',I4//,
     *'NUMBER OF CLASSES OF SUBSTRUCTURES               =',I4//,
     *'MAXIMUM NUMBER OF SUBSTRUCTURE INTER-BOUNDARY'/
     *'NODES IN ANY CLASS                               =',I4//,
     *'NUMBER OF NODES IN MASTER STRUCTURE              =',I4//,
     *'NUMBER OF GLOBAL CASES OF LOADING                =',I4//,
     *'NUMBER OF DEGREES OF FREEDOM/NODE                =',I4//,
     *'NUMBER OF NODES/ELEMENT                          =',I4//,
     *'NUMBER OF GLOBAL EXTERNAL BOUNDARY CONDITIONS    =',I4//,
     *'CONVERGENCE FACTOR OF LENGTH UNIT                =',F12.6/
     */,'DRY RUN FLAG                             ',7X,'=',I4//)
C
      END
```

```
      SUBROUTINE INPUT2(ORINT,NPG,NPL)
C
C   THIS SEGMENT READS THE SUBSTRUCTURE ORIENTATIONS AND THE
C   GLOBAL CONNECTIVITY ARRAY AND THE SUBSTRUCTURE INTER-BOUNDARY
C   NODE NUMBERS ARRAY(NPL).
C
C***********************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      COMMON /MASTIV/ UNIT,NSUB,NCLSUB,MXND,NUMNST,NGLC,IDOF,
     *NODE,MBEL,IDRY,IN,IO
C
      DIMENSION ORINT(1),NPG(MXND,1),NPL(1)
C
      READ(IN,1000) (ORINT(I),I=1,NSUB)
C
      WRITE(IO,2000)
      WRITE(IO,2100) (J,ORINT(J),J=1,NSUB)
C
      WRITE(IO,2200)
      NM  = NSUB/20
      NS1 = 1
      IF(NM.EQ.0) GO TO 20
C
      DO 10 KK=1,NM
      NS2 = KK*NM
      READ(IN,1100) (NPL(J),J=NS1,NS2)
      READ(IN,1100) ((NPG(K,J),J=NS1,NS2),K=1,MXND)
      WRITE(IO,2300) (J,J=NS1,NS2)
      WRITE(IO,2400) (NPL(J),J=NS1,NS2)
      WRITE(IO,2500)
      WRITE(IO,2600) ((NPG(K,J),J=NS1,NS2),K=1,MXND)
      NS1 = NS2 + 1
10    CONTINUE
      IF(NS1.GT.NSUB) GO TO 50
C
20    READ(IN,1100) (NPL(J),J=NS1,NSUB)
      DO 30 K=1,MXND
30    READ(IN,1100) (NPG(K,J),J=NS1,NSUB)
      WRITE(IO,2300) (J,J=NS1,NSUB)
      WRITE(IO,2400) (NPL(J),J=NS1,NSUB)
      WRITE(IO,2500)
      WRITE(IO,2600)
      DO 40 K=1,MXND
40    WRITE(IO,2600) (NPG(K,J),J=NS1,NSUB)
C
50    RETURN
C
C   FORMAT STATEMENTS
C
1000  FORMAT(8F10.0)
1100  FORMAT(20I4)
2000  FORMAT(///T30,'SUBSTRUCTURES OREINTATIONS'//,
     *6(1X,4HNSUB,7X,5HTHETA,3X)//)
2100  FORMAT(6(I5,D15.6))
```

```
2200  FORMAT(//,T30,'GLOBAL CONNECTIVITY DATA'//)
2300  FORMAT(//'SUBSTRUCTURE NUMBER',12X,20I5)
2400  FORMAT(//'NUMBER OF INTER-BOUNDARY NODES',1X,20I5)
2500  FORMAT(//'GLOBAL CONNECTIVITY')
2600  FORMAT(31X,20I5)
C
      END
```

```
      SUBROUTINE INPUT3
C
C  THIS SEGMENT READS CONTROL DATA OF A CLASS OF SUBSTRUCTURES.
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      COMMON /MASTIV/ UNIT,NSUB,NCLSUB,MXNL,NUMNSI,NOLC,IDOF,
     *NODE,MBEL,IDRY,IN,IO
C
      COMMON /SUBIV/ NJ,NE,NLC,NSUBCL,NEBEL,NFIBN,NIBNS,ICLSUB
C
      READ(IN,1000) IC,NJ,NE,NLC,NSUBCL,NEBEL,NFIBN,NIBNS
      IF(IC.NE.ICLSUB) GO TO 999
      WRITE(IO,2000) IC,NJ,NE,NLC,NSUBCL,NEBEL,NFIBN,NIBNS
C
      RETURN
C
999   WRITE(IO,9999) IC,ICLSUB
9999  FORMAT(//,'SUBSTRUCTURE CLASS NUMBER NUMBER ERROR ',2I5)
      STOP
C
C  FORMAT STATEMENTS
C
1000  FORMAT(8I4)
2000  FORMAT('1',T30,'SUBSTRUCTURE CLASS NUMBER ',I5,
     *//,'NUMBER OF JOINTS                                =',I5,
     *//,'NUMBER OF MEMBERS                               =',I5,
     *//,'NUMBER OF LOCAL CASES OF LOADING                =',I5,
     *//,'NUMBER OF SUBSTRUCTURES IN THIS CLASS           =',I5,
     *//,'NUMBER OF EXTERNAL BOUNDARY ELEMENTS            =',I5,
     *//,'LOCAL NUMBER OF FIRST INTER-BOUNDARY NODE       =',I5,
     *//,'NUMBER OF INTER BOUNDARY NODES                  =',I5//)
C
      END
```

```
      SUBROUTINE INPUT4(X,Y,AREA,RI,YMOD,NOD,MKODE,KSUB,NGSUB)
C
C   THIS SEGMENT READS THE GEOMETRY AND THE MEMBER CONNECTIVITY
C   AND PROPERTIES, OF A CLASS OF SUBSTRUCTURES,ALONG WITH THE
C   LOAD CONTROL DATA FOR THE VARIOUS SUBSTRUCTURES THAT BELONG
C   TO THIS CLASS.
C
C*******************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      COMMON /MASTIV/ UNIT,NSUB,NCLSUB,MXND,NOMNST,NGLC,IDOF,
     *NODE,MBEL,IDRY,IN,IO
C
      COMMON /SUBIV/ NJ,NE,NLC,NSUBCL,NEBEL,NFIRN,NIBNS,ICLSUB
C
      DIMENSION X(1),Y(1),AREA(1),RI(1),YMOD(1),NOD(2,1),
     *MKODE(1),KSUB(NGLC,1),NGSUB(1)
C
      WRITE(IO,2000)
50    READ(IN,1000) N,X(N),Y(N),INC
      IF(INC.EQ.0) GO TO 200
      NINT = (N-NOLD)/INC
      RN   = NINT
      IF(RN.LT.FLOAT(N-NOLD)/FLOAT(INC)-0.001) GO TO 999
      DX = (X(N) - X(NOLD))/RN
      DY = (Y(N) - Y(NOLD))/RN
      L = NOLD
      M = NINT - 1
      DO 100 J=1,M
      LL        = L + INC
      X(LL)     = X(L) + DX
      Y(LL)     = Y(L) + DY
      L = LL
100   CONTINUE
200   WRITE(IO,2100) N,X(N),Y(N),INC
      NOLD = N
      IF(N.LT.NJ) GO TO 50
C
      READ(IN,1100) ADEF,RDEF,YDEF
      WRITE(IO,2200) ADEF,RDEF,YDEF
C
      WRITE(IO,2300)
300   READ(IN,1200) M,(NOD(I,M),I=1,2),MKODE(M),INC,NOD1,NOD2,
     *AREA(M),RI(M),YMOD(M)
      IF(AREA(M).EQ.0.) AREA(M)  = ADEF
      IF(RI(M).EQ.0.)   RI(M)    = RDEF
      IF(YMOD(M).EQ.0.) YMOD(M)  = YDEF
      IF(INC.EQ.0) GO TO 500
      NINT = (M-MOLD)/INC - 1
      L    = MOLD
      DO 400 I=1,NINT
      LL          = L + INC
      AREA(LL)    = AREA(M)
      RI(LL)      = RI(M)
      YMOD(LL)    = YMOD(M)
      MKODE(LL)   = MKODE(M)
      NOD(1,LL)   = NOD(1,L)  + NOD1
      NOD(2,LL)   = NOD(2,L)  + NOD2
400   L = LL
C
500   WRITE(IO,2400) M,NOD(1,M),NOD(2,M),MKODE(M),INC,NOD1,NOD2,
     *AREA(M),RI(M),YMOD(M)
      MOLD = M
      IF(M.LT.NE) GO TO 300
C
      READ(IN,1300) K
      IF(K.EQ.0) GO TO 550
      WRITE(IO,2700)
      WRITE(IO,2750)  (N,X(N),Y(N),N=1,NJ)
      WRITE(IO,2800)
      WRITE(IO,2850)  (M,(NOD(I,M),I=1,2),MKODE(M),AREA(M),RI(M),
     *YMOD(M),M=1,NE)
C
550   WRITE(IO,2550)
      NM = NSUBCL/19
      NS1 = 1
      IF(NM.EQ.0) GO TO 800
C
      DO 700 KK=1,NM
      NS2 = KK*19
      READ(IN,1300)  (K,(NGSUB(I),I=NS1,NS2))
      READ(IN,1300)  (K,(KSUB(J,I),I=NS1,NS2),J=1,NGLC)
      WRITE(IO,2500)  (K,K=NS1,NS2)
      WRITE(IO,2650)  (NGSUB(I),I=NS1,NS2)
      DO 600 J=1,NGLC
600   WRITE(IO,2600)  (J,(KSUB(I,J),I=NS1,NS2))
      NS1 = NS2 + 1
700   CONTINUE
C
      IF (NS1.GT.NSUBCL) GO TO 975
C
800   READ(IN,1300)  (K,(NGSUB(I),I=NS1,NSUBCL))
      DO 900 J=1,NGLC
900   READ(IN,1300)  (K,(KSUB(J,I),I=NS1,NSUBCL))
      WRITE(IO,2500)  (K,K=NS1,NSUBCL)
      WRITE(IO,2650)  (NGSUB(I),I=NS1,NSUBCL)
      DO 950 J=1,NGLC
950   WRITE(IO,2600)  (J,(KSUB(J,I),I=NS1,NSUBCL))
C
975   RETURN
C
999   WRITE(IO,9999) N
9999  FORMAT('NODAL GEOMETRY DATA INPUT ERROR',I5)
      STOP
C
C   FORMAT STATEMENTS
C
1000  FORMAT(I4,2F12.0,I4)
1100  FORMAT(3F12.0)
1200  FORMAT(7I4,3F12.0)
```

```
1300  FORMAT(2I4)
2000  FORMAT(//'NODAL GEOMETRY DATA AS INPUT'//,4X,1HN,7X,1HX,
     *14X,1HY,9X,3HINC//)
2100  FORMAT(I5,2D15.6,I5)
2200  FORMAT(//'MEMBER PROPERTIES DEFAULT VALUES'//,
     *'AREA',10X,'=',D15.6,//,
     *'M.INERTIA',5X,'=',D15.6,//,
     *'Y.MODULUS',5X,'=',D15.6)
2300  FORMAT(//'MEMBER DATA AS INPUT'//,
     *4X,1HM,4X,1HI,4X,1HJ,1X,4HCODE,2X,3HINC,1X,4HINCI,1X,
     *4HINCJ,5X,4HAREA,10X,1HI,13X,4HYMOD//)
2400  FORMAT(7I5,3D15.6)
2500  FORMAT(//,'L.SUBSTRUCTURE NO.   ',13I5//)
2550  FORMAT('1','LOAD CASES CONTROL MATRIX')
2600  FORMAT('G.LOAD CASE(',I4,')    ',19I5)
2650  FORMAT('G.SUBSTRUCTURE NO.   ',19I5)
2700  FORMAT('1','COMPLETED NODAL GEOMETRY DATA'//,
     *4X,1HN,7X,1HX,14X,1HY//)
2750  FORMAT(I5,2D15.6)
2800  FORMAT('1','COMPLETED MEMBER DATA',//,
     *4X,1HM,4X,1HI,4X,1HJ,1X,4HCODE,5X,4HAREA,10X,1HI,13X,
     *4HYMOD//)
2850  FORMAT(4I5,3D15.6)
C
      END
```

C.2 Program MUSAPF

Program MUSAPF consists of the following parts.

1. MAIN
2. MAINMG, the main executive subroutine.
3. The input package

   INPUT1, INPUT2, INPUT3, INPUT4, INPUT5, INPUT6.
4. The data managing package

   ISPAC, LOCOM, REMOV, REMOV2, and BLOCK DATA.
5. The data storage and retrieval package

   CLEAR, ICLEAR, RTRV1, RTRV2, STORE1, STORE2.
6. The equation solving package

   ADDRES, COLHT, EQSBST, EQFT, EQKBB, BKSB1, BKSB2.
7. The formulation and output package

   ASSEMB, BOUND, BOUND2, DISPL, JLOAD, LOADID, MLLOADS, RESUB, STIFF, and STRESS.

The listing for the first three parts follows. The listing for parts 4, 5, 6, and 7 can be found in Sections C.3, C.4, C.5, and C.6, respectively.

The modifications discussed in Sections 3.4.2, and 3.4.3 have been implemented by adding subroutine MODAX1 to the equation solving package for modification 1, and by subroutine SKYPRD to the same package for modification 2. The listing and description of both subroutines can be found in Section C.5. The corresponding changes to the main executive subroutine MAINMG are not shown herein.

```
C*************************************************************************
C
C                          MUSAPP
C
C   THIS IS A PROGRAM FOR MULTI-LEVEL SUBSTRUCTURE ANALYSIS OF
C   PLANER FRAMED TYPE STRUCTURES. NO LIMITATIONS ARE PLACED
C   ON THE NUMBER OF, OR THE NUMBER OF LEVELS OF SUBSTRUCTURES,
C   OR THE NUMBER OF CASES OF LOADING. THE EQUATION SOLVER IS
C   OF SKYLINE IN-CORE TYPE SOLVER.
C
C*************************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 NAMES
C
      COMMON/PROBIA/ NNN(300)
      COMMON/IA1/    MMM(1500)
      COMMON/IA2/    KKK(3000)
      COMMON/RA1/    AAA(7000)
      COMMON/RA2/    BBB(35000)
      COMMON /DIMCOM/ LAST1,LAST2,LAST3,LAST4,LAST5,MXDIM
     *,NAMES(5,20),IPT(5,21),ICOM(5)
C
C*************************************************************************
C
      ICOM(1) = 300
      ICOM(2) = 7000
      ICOM(3) = 1500
      ICOM(4) = 35000
      ICOM(5) = 3000
C
      CALL MAINMG
C
      END
```

```
      SUBROUTINE MAINMG
C
C   THIS IS THE MAIN MANAGER OF PROGRAM MUSAPP.
C   IT CONTROLS THE READING, THE FORMULATION, THE SOLUTION, AND
C   OUTPUT OF THE RESULTS.
C
C*************************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 NAMES,NAME
      COMMON/PROBCV/UNIT,NSUB,NINDSB,NTOTAS,NINDAS,NGLC,IDOF,
     *NODE,IN,IO,NMBKSB,IDRY
      COMMON/HLCV1/ ICH,NJH,NLCH,NEBELH,NFIBNH,NIBNSH
      COMMON/HLCV2/ NCU,MXND,IFLAG
      COMMON/LLCV/ ICLSUB,NJ,NLC,NEBEL,NFIBN,NIBNS,NE
      COMMON/PROBIA/ NNN(1)
      COMMON/IA1/    MMM(1)
      COMMON/IA2/    KKK(1)
      COMMON/RA1/    AAA(1)
      COMMON/RA2/    BBB(1)
      COMMON /DIMCOM/ LA1,LA2,LA3,LA4,LA5,MXDIM,NAMES(5,20),
     *IPT(5,21),ICOM(5)
C
      DIMENSION INFO(4)
C
C*************************************************************************
      CALL TIME(0,0)
      IN = 5
      IO = 6
      CALL CLEAR(NAMES(1,1),100)
      REWIND 1
      REWIND 2
      REWIND 3
C
C   READ PROBLEM CONTROL VARIABLES
      CALL INPUT1
C
      IF(NTOTAS.EQ.1) GO TO 10
C
C   READ BACKSUBSTITUTION INFORMATION ARAAYS
      I1 = ISPAC(6HINF1BK,NMBKSB,1)
      I2 = ISPAC(6HINF2BK,NMBKSB,1)
      I3 = ISPAC(6HINF3BK,NMBKSB,1)
      I4 = ISPAC(6HINF4BK,NMBKSB,1)
      CALL INPUT2(NNN(I1),NNN(I2),NNN(I3),NNN(I4))
C
      I5 = ISPAC(6HIRTRV1,NTOTAS,1)
      I6 = ISPAC(6HIRTRV2,NTOTAS,1)
      I7 = ISPAC(6HIRTRV3,NTOTAS,1)
      CALL TIME(3,3)
      WRITE(IO,2000)
C
C   START TO LOOP OVER INDEPENDENT BASIC UNITS.
10    ICLSUB = 1
C
C   READ INDEPENDENT BASIC UNIT(ICLSUB) CONTROL VARIABLES.
```

```
20    CALL INPUT3
C
C   READ INDEPENDENT BASIC UNIT(ICLSUB) NODAL GEOMETRY AND MEMB-
C   ER PROPERTIES, AND CONNECTIVITIES.
      J1 = ISPAC(1HX,NJ,2)
      J2 = ISPAC(1HY,NJ,2)
      J3 = ISPAC(4HAREA,NE,2)
      J4 = ISPAC(2HRI,NE,2)
      J5 = ISPAC(4HYMOD,NE,2)
      K1 = ISPAC(3HNOD,(NE*NODE),3)
      K2 = ISPAC(5HMKODE,NE,3)
      CALL INPUT4(AAA(J1),AAA(J2),AAA(J3),AAA(J4),AAA(J5),
     *MMM(K1),MMM(K2))
C
      J6 = ISPAC(3HFEF,(IDOF*NODE*NE*NLC),2)
      L1 = ISPAC(1HB,(IDOF*NJ*NLC),4)
      CALL CLEAR(AAA(J6),(IDOF*NODE*NE*NLC))
      CALL CLEAR(BBB(L1),(IDOF*NJ*NLC))
      IF(IDRY.EQ.1) GO TO 25
C
C   READ EXTERNAL BOUNDARY CONDITIONS IF ANY.
25    IF(NEBEL.EQ.0) GO TO 30
      J7 = ISPAC(8HBES     ,NEBEL,2)
      K3 = ISPAC(3HNPB,NEBEL,3)
      K4 = ISPAC(4HKODE,NEBEL,3)
      CALL BOUND(AAA(J7),MMM(K3),MMM(K4),NEBEL,IN,IO)
      GO TO 40
30    J7 = 1
C
40    WRITE(IO,2100) ICLSUB
C
C   READ LOADS IF ANY.
C
      READ(IN,1000) JLFLAG,MLFLAG
C
C   READ JOINT LOADS AND OR DISPLACEMENTS IF ANY.
      IF(JLFLAG.EQ.0) GO TO 50
      K5 = ISPAC(5HJKODE,NLC,3)
      CALL JLOAD(MMM(K5),BBB(L1),AAA(J7),NLC,IN,IO)
      CALL REMOV(5HJKODE,3)
C
C   READ AND PREPARE MEMBER LOADS IF ANY.
50    IF(MLFLAG.EQ.0) GO TO 60
      CALL MLOADS(AAA(J1),AAA(J2),AAA(J6),BBB(L1),MMM(K1),
     *MMM(K2),IDRY,UNIT,IN,IO,NLC,NJ)
C
60    WRITE(IO,2200) ICLSUB
C
C   FORM COLUMN HEIGHTS AND ADDRESSING ARRAY.
      M1 = ISPAC(4HMAXA,(IDOF*NJ+1),5)
      M2 = ISPAC(3HMHT,(IDOF*NJ),5)
      M3 = ISPAC(2HLM,(IDOF*NODE),5)
      CALL ICLEAR(KKK(M2),(IDOF*NJ))
      DO 70 I=1,NE
      CALL COLHT(NODE,NODE,(NODE*IDOF),IDOF,I,KKK(M2),MMM(K1),
     *KKK(M3))
70    CONTINUE
      CALL ADDRES(KKK(M1),KKK(M2),NJ,IDOF,NWA)
      CALL REMOV(2HLM,5)
      CALL REMOV(3HMHT,5)
      IF(IDRY.EQ.1) GO TO 80
C
C   RESERVE AND CLEAR SPACE FOR STIFFNESS MATRIX
      L2 = ISPAC(1HA,NWA,4)
      CALL CLEAR(BBB(L2),NWA)
C
C   FORM AND ASSEMBLE ELEMENT STIFFNESSES INTO A.
      CALL STIFF(MMM(K1),MMM(K2),KKK(M1),AAA(J1),AAA(J2),AAA(J3)
     *,AAA(J4),AAA(J5),BBB(L2),UNIT,NE)
C
C   ADD EXTERNAL BOUNDARY CONDITIONS IF ANY.
      IF(NEBEL.EQ.0) GO TO 80
      CALL BOUND2(BBB(L2),KKK(M1),MMM(K3),MMM(K4),AAA(J7),NEBEL)
      CALL REMOV(4HKODE,3)
      CALL REMOV(3HNPB,3)
      CALL REMOV(8HBES     ,2)
C
80    WRITE(IO,2300) ICLSUB,NWA
      CALL TIME(3,3)
      IF(IDRY.EQ.1) GO TO 95
C
C   REDUCE A AND B IF REQUIRED, TO LEVEL OF PARTITION, IF ANY.
      NI = NJ - NIBNS
      IF(NI.EQ.0) GO TO 90
      CALL EQSBST(BBB(L2),BBB(L1),KKK(M1),NI,IDOF,NLC)
      IF(NTOTAS.EQ.1) GO TO 100
      CALLEQFT(BBB(L2),KKK(M1),NI,IDOF,NJ)
      CALL EQKBB(BBB(L2),BBB(L1),KKK(M1),NI,IDOF,NJ,NLC)
      CALL TIME(3,3)
C
C   WRITE THE RELEVANT INFORMATION ON FILES 1 AND 2.
90    CALL NOTE(1,INFO)
      NNN(I5+ICLSUB-1) = INFO(2)
      CALL NOTE(2,INFO)
      NNN(I6+ICLSUB-1) = INFO(2)
C
      II3 = IPT(3,LA3+1) - 1
      II2 = IPT(2,LA2+1) - 1
      II5 = IPT(5,LA5+1) - 1
      II4 = NJ*NLC*IDOF
C
      WRITE(1) ICLSUB,J1,J2,J3,J4,J5,J6,NE,II3,II2
      CALL STORE1(1,MMM(1),II3)
      CALL STORE2(1,AAA(1),II2)
      WRITE(2) ICLSUB,NJ,NLC,NFIBN,NIBNS,NWA,II5,II4
      CALL STORE2(2,BBB(1),II4)
      CALL STORE2(2,BBB(L2),NWA)
      CALL STORE1(2,KKK(1),II5)
      WRITE(IO,2500) ICLSUB
C
```

```
95    CALL REMOV2(2)
      CALL REMOV2(3)
      CALL REMOV2(4)
      CALL REMOV2(5)
C
      CALL TIME(3,3)
C
      ICLSUB = ICLSUB+ 1
      IF(ICLSUB.LE.NINDSB) GO TO 20
      GO TO 120
C
C*****************************************************************
C
C   THIS IS A BRANCH TO AN UNSUBSTRUCTURED PROBLEM. PROCEED WITH
C   BACKSUBSTITUTION FROM THE TRINGULARIZED STIFFNESS MATRIX.
C   OUTPUT DISPLACEMENTS AND MEMBER END FORCES
C
100   CALL BKSB1(BBB(L1),BBB(L2),KKK(M1),NI,NLC,IDOF,0)
C
      CALL TIME(3,3)
      WRITE(IO,2600)
      L3 = ISPAC(4HFEF2,(IDOF*NODE*NE),4)
      DO 110 I=1,NLC
      CALL DISPL(BBB(L3),BBB(L1),AAA(J6),MMM(K1),I,1,1,NJ,NE,NLC
     *,NLC,IN,IO,NTOTAS)
C
      CALL  STRESS(AAA(J1),AAA(J2),AAA(J3),AAA(J4),AAA(J5),
     *BBB(L1),BBB(L3),MMM(K1),MMM(K2),NE,I,NLC,UNIT,IN,IO)
C
110   CONTINUE
C
C   END OF PROBLEM.
C
115   CALL TIME(3,3)
      RETURN
C
C*****************************************************************
C
C   START TO LOOP OVER ASSEMBLIES OF MORE THAN ONE UNIT.
C
C   READ ASSEMBLY UNIT(ICLSUB) CONTROL VARIABLES
120   IF(NTOTAS.EQ.1.AND.IDRY.EQ.1) GO TO 115
      ICH = ICLSUB
      CALL INPUT5
C
C   READ ASSEMBLY UNIT(ICLSUB)'S CONSTITUENT UNITS INFORMATION .
      L0 = ISPAC(5HORINT,NCU,4)
      K1 = ISPAC(6HINDSUB,NCU,3)
      K2 = ISPAC(6HIRCSUB,NCU,3)
      K3 = ISPAC(6HIBNSUB,NCU,3)
      K4 = ISPAC(3HNPG,(NCU*MXND),3)
      K5 = ISPAC(5HLCSUB,(NCU*NLCH),3)
      CALL INPUT6(BBB(L0),MMM(K1),MMM(K2),MMM(K3),MMM(K4),
     *MMM(K5))
C
C   READ UNIT(ICLSUB) EXTERNAL BOUNDARY CONDITIIPNS, IF ANY.
      IF(NEBELH.EQ.0) GO TO 130
      J1 = ISPAC(3HBES,NEBELH,2)
      K6 = ISPAC(3HNPB,NEBELH,3)
      K7 = ISPAC(4HKODE,NEBELH,3)
      CALL BOUND(AAA(J1),MMM(K6),MMM(K7),NEBELH,IN,IO)
C
130   WRITE(IO,2100) ICLSUB
C
C   FORM COLUMN HIGHTS AND ADDRESSING ARRAY.
      M1 = ISPAC(4HMAXB,(NJH*IDOF+1),5)
      M2 = ISPAC(3HMHB,(IDOF*NJH),5)
      CALL ICLEAR(KKK(M2),(NJH*IDOF))
      M3 = ISPAC(2HLM,(IDOF*MXND),5)
C
      DO 140 I=1,NCU
      NODES = MMM(K3-1+I)
      ND    = NODES*IDOF
      CALL COLHT(MXND,NODES,ND,IDOF,I,KKK(M2),MMM(K4),KKK(M3))
140   CONTINUE
      CALL ADDRES(KKK(M1),KKK(M2),NJH,IDOF,NWK)
      CALL REMOV(2HLM,5)
      CALL REMOV(3HMHB,5)
      IF(IDRY.EQ.1) GO TO 170
C
C   RESERVE AND CLEAR SPACE FOR STIFFNESS AND LOAD ARRAYS.
      L1 = ISPAC(2HBP,(NJH*IDOF*NLCH),4)
      L2 = ISPAC(2HBB,(NWK),4)
      CALL CLEAR(BBB(L1),(NJH*IDOF*NLCH+NWK))
C
C   ASSEMBLE STIFFNESS AND LOAD VECTOR OF UNIT(ICLSUB).
      DO 160 I=1,NCU
      IND = MMM(K1-1+I)
      IRC = MMM(K2-1+I)
      IF(IRC.EQ.0) GO  TO 150
      INFO(1) = MMM(I6-1+IND)
      CALL POINT(2,INFO,1)
      READ(2) ICL,NJ,NLC,NFIBN,NIBNS,NWA,II3,II4
      L3 = ISPAC(1HB,(NJ*IDOF*NLC),4)
      L4 = ISPAC(1HA,NWA,4)
      M4 = ISPAC(4HMAXA,(NJ*IDOF+1),5)
      CALL RTRV2(2,BBB(L3),II4)
      CALL RTRV2(2,BBB(L4),NWA)
      CALL RTRV1(2,KKK(M4),II3)
150   CALL ASSEMB(MMM(K4),KKK(M1),MMM(K5),KKK(M4),BBB(L0),BBB(L1
     *),BBB(L2),BBB(L3),BBB(L4),I,I,NFIBN,NIBNS,MXND,NLC,NLCH)
      IF(IRC.EQ.0) GO TO 160
      CALL REMOV(4HMAXA,5)
      CALL REMOV(1HA,4)
      CALL REMOV(1HB,4)
160   CONTINUE
C
C   ADD BOUNDARY CONDITIONS IF ANY, TO STIFFNESS MATRIX.
      IF(NEBELH.EQ.0) GO TO 170
      CALL BOUND2(BBB(L2),KKK(M1),MMM(K6),MMM(K7),AAA(J1),NEBELH)
```

```
      CALL REMOV(4HKODE,3)
      CALL REMOV(3HNPB,3)
170   CALL TIME(3,3)
      WRITE(IO,2300) ICLSUB,NWK
      IF(IDRY.EQ.1) GO TO 185
C
C   REDUCE STIFFNESS MATRIX AND LOAD VECTOR IF ANY TO LEVEL OF
C   PARTITION IF REQUIRED.
      NI = NJH - NIBNSH
      IF (NI.EQ.0) GO TO 180
      CALL EQSBST(BBB(L2),BBB(L1),KKK(M1),NI,IDOF,NLCH)
      IF(ICLSUB.EQ.NINDAS) GO TO 190
      CALL EQPT(BBB(L2),KKK(M1),NI,IDOF,NJH)
      CALL EQKBB(BBB(L2),BBB(L1),KKK(M1),NI,IDOF,NJH,NLCH)
      CALL TIME(3,3)
C
C   STORE THE RELEVANT INFORMATION OF UNIT(ICLSUB) ON FILES 1,2.
180   CALL NOTE(1,INFO)
      NNN(I5-1+ICLSUB) = INFO(2)
      CALL NOTE(2,INFO)
      NNN(I6-1+ICLSUB) = INFO(2)
C
      II1 = IPT(3,LA3+1) - 1 - 3*NCU
      II2 = NCU
      II3 = IPT(5,LA5+1) -1
      II4 = NLCH*NJH*IDOF
C
      WRITE(1) ICLSUB,NJH,NCU,MXND,NLCH,II2,II1
      CALL STORE2(1,BBB(1),II2)
      CALL STORE1(1,MMM(K4),II1)
      WRITE(2) ICLSUB,NJH,NLCH,NFIBNH,NIBNSH,NWK,II3,II4
      CALL STORE2(2,BBB(L1),II4)
      CALL STORE2(2,BBB(L2),NWK)
      CALL STORE1(2,KKK(1),II3)
      WRITE(IO,2500) ICLSUB
C
185   CALL REMOV2(3)
      CALL REMOV2(4)
      CALL REMOV2(5)
      IF(NEBELH.EQ.0) GO TO 187
      CALL REMOV(3HBBS,2)
C
187   CALL TIME(3,3)
      IF(IDRY.EQ.1.AND.ICLSUB.EQ.NINDAS) GO TO 340
      ICLSUB = ICLSUB + 1
      IF(ICLSUB.GT.NINDAS) GO TO 991
      GO TO 120
C
C***************************************************************
C
C   GET THE SOLUTION VECTOR FOR MASTER SYSTEM.
C
190   CALL BKSB1(BBB(L1),BBB(L2),KKK(M1),NI,NGLC,IDOF,0)
C
C   STORE MASTER SYSTEM SOLUTION IF REQUIRED.
      IF(IFLAG.EQ.0) GO TO 200
      CALL NOTE(3,INFO)
      NNN(I7-1+NTOTAS) = INFO(2)
      II1 = NGLC*NJH*IDOF
      II2 = NGLC*NCU
      WRITE(3) NTOTAS,NINDAS,II1,II2
      CALL STORE2(3,BBB(L1),II1)
      CALL STORE1(3,MMM(K5),II2)
C
      CALL NOTE(1,INFO)
      NNN(I5-1+NINDAS) = INFO(2)
      II1 = NCU*MXND
      II2 = NCU
      WRITE(1) ICLSUB,NJH,NCU,MXND,NLCH,II2,II1
      CALL STORE2(1,BBB(1),II2)
      CALL STORE1(1,MMM(K4),II1)
C
200   CALL REMOV(2HBB,4)
      CALL REMOV2(5)
C
      CALL TIME(3,3)
      WRITE(IO,2700)
C
C***************************************************************
C
C   START THE OUTPUT GOVERNNING LOOP.
C
      I=1
      ISUB1 = NTOTAS
      ISUB2 = 0
      INF1  = NNN(I1-1+I)
      INF2  = NNN(I2-1+I)
      INF3  = NNN(I3-1+I)
      INF4  = NNN(I4-1+I)
C
C   READ HIGHER UNIT(INF1) SOLUTION VECTOR, AND RESUBSTITUTION
C   INFORMATION FROM FILE 3,IF IT IS NOT IN CORE AT THIS STAGE.
210   IF(INF1.EQ.ISUB1) GO TO 220
      INFO(1) = NNN(I7-1+INF1)
      CALL POINT(3,INFO,1)
      READ(3) IGH,IND,II3,II4
      IF(IGH.NE.INF1) GO TO 992
      INFO(1) = NNN(I5-1+IND)
      CALL POINT(1,INFO,1)
      READ(1) ICH,NJH,NCU,MXND,NLCH,II1,II2
      IF(ICH.NE.IND) GO TO 993
      K4 = ISPAC(3HNPG,(NCU*MXND),3)
      K5 = ISPAC(4HKSUB,II4,3)
      L0 = ISPAC(5HORINT,NCU,4)
      L1 = ISPAC(2HBP,II3,4)
      CALL RTRV2(3,BBB(L1),II3)
      CALL RTRV1(3,MMM(K5),II4)
      CALL RTRV2(1,BBB(L0),II1)
      CALL RTRV1(1,MMM(K4),(MXND*NCU))
C
```

```
C
C    READ LOWER UNIT(INF3) INFORMATION FROM FILE 2,IF IT IS NOT
C    IN CORE AT THIS STAGE.
220    IF(INF3.EQ.ISUB2) GO TO 230
       INFC(1) = NNN(I6-1+INF3)
       CALL PCINT(2,INFC,1)
       READ(2)ICL,NJ,NLC,NFIBN,NIBNS,NWA,II3,II4
       L3 = ISPAC(8HB         ,II4,4)
       L4 = ISPAC(2HBA,(NGLC*NJ*IDCF),4)
       CALL RTRV2(2,BBB(L3),II4)
C
C    RESUBSTITUTE FROM SOLUTION VECTOR OF INF1 INTO DUMMY VECTOR
C    BA OF SYSTEM INF2.
C
230    CALL RESUB(BBB(L3),BBB(L4),BBB(L5),BBB(L1),MMM(K4),MMM(K5)
      *,INF4,INF4,NGLC,MXND,NFIBN,NIBNS,NLC)
C
C    BACKSUBSTITUTE FROM STIFFNESS MATRIX OF SYSTEM INF3 INTO
C    DUMMY VECTOR BA OF SYSTEM INF2.
       NI = NJ - NIBNS
       IF(NI.EQ.0) GO TO 250
       IF(INF3.EQ.ISUB2) GO TO 240
       L5 = ISPAC(1HA,NWA,4)
       CALL RTRV2(2,BBB(L5),NWA)
       M1 = ISPAC(4HMAXA,(NJ*IDOF+1),5)
       CALL RTRV1(2,KKK(M1),II3)
240    CALL BKSB2(BBB(L5),BBB(L4),KKK(M1),NFIBN,NJ,NGLC,IDOF)
       CALL BKSB1(BBB(L4),BBB(L5),KKK(M1),NI,NGLC,IDOF,1)
C
C    CHECK IF OUTPUT IS REQUIRED HERE.
250    IF(INF2.GT.NSUB) GO TO 280
C
C    SYSTEM INF2 IS A BASIC OUTPUT UNIT. CONTINUE WITH OUTPUT
C    OPERATIONS.
       IF(INF3.EQ.ISUB2)  GO TO 260
       INFO(1) = NNN(I5-1+INF3)
       CALL PCINT(1,INFO,1)
       READ(1) ICL,J1,J2,J3,J4,J5,J6,NE,II1,II2
       IF(ICL.NE.INF3) GC TO 994
C
       M2 = ISPAC(3HNOD,(NE*NODE),5)
       M3 = ISPAC(5HMKODE,NE,5)
       CALL RTRV1(1,KKK(M2),II1)
       CALL RTRV2(1,AAA(J1),II2)
       L7 = ISPAC(4HFEF2,(NE*NODE*IDOF),4)
C
260    DO 270 J=1,NGLC
       CALL DISPL(BBB(L7),BBB(L4),AAA(J6),MMM(K5),J,INF4,INF2,NJ
      *,NE,NLC,NGLC,IN,IC,NTOTAS)
C
       CALL STRESS(AAA(J1),AAA(J2),AAA(J3),AAA(J4),AAA(J5),
      *BBB(L4),BBB(L7),KKK(M2),KKK(M3),NE,J,NGLC,UNIT,IN,IO)
C
270    CONTINUE
       GO TO 300
C
C    SYSTEM INF2 IS AN INTERMEDIATE UNIT. FORM KSUB FOR THIS
C    SYSTEM, AND STORE SOLUTION VECTOR BA AND LOAD ID. ARRAY KSUB
C    ON FILE 3.
280    IF(INF3.EQ.ISUB2) GO TO 290
       INFC(1) = NNN(I5-1+INF3)
       CALL PCINT(1,INFC,1)
       READ(1) ICL,NJ,NCUL,MXNDL,NLC,II1,II2
       J8 = ISPAC(5HORINT,NCUL,2)
       M2 = ISPAC(3HNPG,(MXNDL*NCUL),5)
       M3 = ISPAC(5HLCSUB,(NLC*NCUL),5)
       M4 = ISPAC(4HKSUB,(NCUL*NGLC),5)
       CALL RTRV2(1,AAA(J8),II1)
       CALL RTRV1(1,KKK(M2),II2)
290    CALL LCADID (MMM(K5),KKK(M3),KKK(M4),INF4,NCUL,NGLC,NLC)
C
       CALL NCTE(3,INFO)
       NNN(I7-1+INF2) = INFO(2)
       II1 = NGLC*NJ*IDCF
       II2 = NGLC*NCUL
       WRITE(3) INF2,INF3,II1,II2
       CALL STORE2(3,BBB(L4),II1)
       CALL STORE1(3,KKK(M4),II2)
C
C    CHECK AND UPDATE LOOP INDICES.
300    CALL TIME(3,3)
       WRITE(IC,2800) I
       I = I + 1
C
       IF(I.GT.NMBKSB) GO TO 340
       ISUB2 = INF3
       ISUB1 = INF1
       INF1 = NNN(I1-1+I)
       INF2 = NNN(I2-1+I)
       INF3 = NNN(I3-1+I)
       INF4 = NNN(I4-1+I)
       IF(INF3.EQ.ISUB2) GO TO 320
       CALL REMCV2(5)
       CALL REMOV2(2)
       IF(ISUB2.GT.NINDSB) GO TO 310
       CALL REMOV(4HFEF2,4)
310    IF(NI.EQ.0) GO TO 315
       CALL REMOV(1HA,4)
315    CALL REMCV(2HBA,4)
       CALL REMOV(8HB         ,4)
C
320    IF(INF1.EQ.ISUB1) GO TO 330
       CALL REMCV2(3)
       CALL REMOV2(4)
C
330    GO TO 210
C
340    RETURN
C
C    END CF PROBLEM.
```

```
C*********************************************************************
C
C   ERROR DIAGNOSTICS' FORMAT.
991   WRITE(IO,3000)
      GO TO 400
992   WRITE(IO,3100)  IGH,INF1
      GO TO 400
993   WRITE(IO,3200)  ICH,IND
      GO TO 400
994   WRITE(IO,3300)  ICL,INF3
400   STOP
C*********************************************************************
C
C   FORMAT STATEMENTS
1000  FORMAT(2I4)
2000  FORMAT(//,'GLOBAL PARAMETERS INPUT IS COMPLETED')
2100  FORMAT(//,'STRUCTURAL DATA INPUT OF UNIT(',I4,
     *')IS COMPLETED')
2200  FORMAT(//,'LOAD VECTOR OF UNIT(',I4,') IS FORMED')
2300  FORMAT(//,'STIFFNESS MATRIX OF UNIT(',I4,') IS FORMED.'/
     *' NWA(NWK) =',I6)
2500  FORMAT(//,'UNIT(',I4,'HAS BEEN COMPLETELY DEFINED,REDUCED'
     *,', AND STORED.')
2600  FORMAT(//,'BACKSUBSTITUTION OF THE UNSUBSTRUCTURED '
     *,'PROBLEM IS COMPLETED')
2700  FORMAT(//,'SOLUTION VECTOR OF MASTER SYSTEM IS COMPLETED')
2800  FORMAT(//,'BACK SUBSTITUTION STEP NUMBER(',I4,
     *') IS COMPLETED.')
3000  FORMAT(//,'MASTER SYSTEM FLAG ERROR')
3100  FORMAT(//,'BACKSUBSTITUTION ERROR,IGH=',I4,10X,'INF1=',I4)
3200  FORMAT(//,'BACKSUBSTITUTION ERROR,ICH=',I4,10X,'IND=',I4)
3300  FORMAT(//,'BACKSUBSTITUTION ERROR,ICL=',I4,10X,'INF3=',I4)
C
      END
```

```
      SUBROUTINE INPUT1
C
C   THIS SEGMENT READS THE PROBLEM MACRO CONTROL VARIABLES FOR
C   PROGRAM MUSAPP.
C
C*****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON/PROBCV/UNIT,NSUB,NINDSB,NTOTAS,NINDAS,NGLC,IDOF,
     *NODE,IN,IO,NMBKSB,IDRY
C
      DIMENSION HED(20)
C
      READ(IN,1000) HED
      WRITE(IO,2000) HED
C
      READ(IN,1100) NSUB,NINDSB,NTOTAS,NINDAS,NGLC,NMBKSB,IDOF,
     *NODE,UNIT,IDRY
      WRITE(IO,2100) NSUB,NINDSB,NTOTAS,NINDAS,NGLC,NMBKSB,IDOF,
     *NODE,UNIT,IDRY
C
      RETURN
C
C   FORMAT STATEMENTS
C
1000  FORMAT(20A4)
1100  FORMAT(8I4,F12.0,I4)
2000  FORMAT(20X,20A4///)
2100  FORMAT('PROBLEM CONTROL VARIABLES'//,25(1H*)//,
     *'TOTAL NUMBER OF BASIC SUBSTRUCTURES              =',I4//,
     *'TOTAL NUMBER INDEPENDENT BASIC SUBSTRUCTURES     =',I4//,
     *'TOTAL NUMBER OF SUBSTRUCTURES AND/OR ASSEMBLIES'/
     *'OF SUBSTRUCTURES, UP TO THE MASTER SYSTEM        =',I4//,
     *'TOTAL NUMBER OF INDEPENDENT SUBSTRUCTURES        =',I4//,
     *'TOTAL NUMBER OF GLOBAL CASES OF LOADING          =',I4//,
     *'TOTAL NUMBER OF BACKSUBSTITUTION STEPS           =',I4//,
     *'NUMBER OF DEGREES OF FREEDOM/NODE                =',I4//,
     *'NUMBER OF NODES/ELEMENT                          =',I4//,
     *'LENGTH CONVERGENCE FACTOR                        =',F12.6//,
     *'DRY RUN FLAG                                     =',I4//,
     *80(1H*))
C
      END
```

```
      SUBROUTINE INPUT2(INF1BK,INF2BK,INF3BK,INF4BK)
C
C   THIS SUBROUTINE READS THE INFORMATION NECESSARY FOR CONTROL
C   OF BACKSUBSTITUTION. PROGRAMM MUSAPF.
C
C*****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON/PROBCV/UNIT,NSUB,NINDSB,NTOTAS,NINDAS,NGLC,IDOF,
     *NODE,IN,IO,NMBKSB,IDRY
C
      DIMENSION INF1BK(1),INF2BK(1),INF3BK(1),INF4BK(1)
C
      WRITE(IO,2000)
      NM = NMBKSB/20
      NS1 = 1
      IF(NM.EQ.0) GO TO 20
C
      DO 10 KK=1,NM
      NS2 = KK*20
      READ(IN,1000)  (INF1BK(I),I=NS1,NS2)
      READ(IN,1000)  (INF2BK(I),I=NS1,NS2)
      READ(IN,1000)  (INF3BK(I),I=NS1,NS2)
      READ(IN,1000)  (INF4BK(I),I=NS1,NS2)
      WRITE(IO,2100)  (K,K=NS1,NS2)
      WRITE(IO,2200)  (INF1BK(I),I=NS1,NS2)
      WRITE(IO,2300)  (INF2BK(I),I=NS1,NS2)
      WRITE(IO,2400)  (INF3BK(I),I=NS1,NS2)
      WRITE(IO,2500)  (INF4BK(I),I=NS1,NS2)
      NS1 = NS2 + 1
10    CONTINUE
      IF(NS1.GT.NMBKSB) GO TO 30
C
20    READ(IN,1000)  (INF1BK(I),I=NS1,NMBKSB)
      READ(IN,1000)  (INF2BK(I),I=NS1,NMBKSB)
      READ(IN,1000)  (INF3BK(I),I=NS1,NMBKSB)
      READ(IN,1000)  (INF4BK(I),I=NS1,NMBKSB)
      WRITE(IO,2100)  (K,K=NS1,NMBKSB)
      WRITE(IO,2200)  (INF1BK(I),I=NS1,NMBKSB)
      WRITE(IO,2300)  (INF2BK(I),I=NS1,NMBKSB)
      WRITE(IO,2400)  (INF3BK(I),I=NS1,NMBKSB)
      WRITE(IO,2500)  (INF4BK(I),I=NS1,NMBKSB)
C
30    RETURN
C
C   FORMAT STATEMENTS
1000  FORMAT(20I4)
2000  FORMAT(//'BACKSUBSTITUTION INFORMATION'/,28(1H*))
2100  FORMAT(//'STEP NUMBER                            ',20I4)
2200  FORMAT(/'HIGER LEVEL UNIT GLOBAL NUMBER          ',20I4)
2300  FORMAT('LOWER LEVEL UNIT GLOBAL NUMBER           ',20I4)
2400  FORMAT('LOWER LEVEL UNIT INDPENDENT NUMBER       ',20I4)
2500  FORMAT('L.L.U.POSITION IN H.L.U. ARRAYS          ',20I4)
C
      END
```

```
      SUBROUTINE INPUT3
C
C   THIS SUBROUTINE READS INDPENDENT BASIC UNIT(ICLSUB)  CONTROL
C   VARIABLES. PROGRAMM MUSAPP.
C
C*****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON/PROBCV/UNIT,NSUB,NINDSB,NTOTAS,NINDAS,NGLC,IDOF,
     *NODE,IN,IO,NMBKSB,IDRY
C
      COMMON /LLCV/ ICLSUB,NJ,NLC,NEBEL,NFIBN,NIBNS,NE
C
      READ(IN,1000) IC,NJ,NE,NLC,NEBEL,NFIBN,NIBNS
      IF(IC.NE.ICLSUB) GO TO 999
      WRITE(IO,2000) IC,NJ,NE,NLC,NEBEL,NFIBN,NIBNS
C
      RETURN
C
999   WRITE(IO,9999) IC , ICLSUB
9999  FORMAT('BASIC UNITS ARE NOT ENTERED IN PROPER ORDER',2I5)
      STOP
C
C   FORMAT STATEMENTS
C
1000  FORMAT(7I4)
2000  FORMAT('1','INDPENDENT BASIC UNIT(',I4,')',/26(1H*)//,
     *'CONTROL VARIABLES'//,
     *'TOTAL NUMBER OF NODES                             =',I4//,
     *'TOTAL NUMBER OF ELEMENTS                          =',I4//,
     *'TOTAL NUMBER OF INDPENDENT CASES OF LOADING       =',I4//,
     *'TOTAL NUMBER OF EXTERNAL BOUNDARY ELEMENTS        =',I4//,
     *'LOCAL NUMBER OF FIRST INTER-BOUNDARY NODE         =',I4//,
     *'TOTAL NUMBER OF INTER-BOUNDARY NODES              =',I4//)
C
      END
```

```
      SUBROUTINE INPUT4(X,Y,AREA,RI,YMOD,NOD,MKODE)
C
C   THIS SUBROUTINE READS THE NODAL GEOMETRY,AND MEMBER PROPERT-
C   IES AND CONNECTIVITIES,FOR INDPENDENT BASIC UNIT(ICLSUB).
C   PROGRAMM MUSAPP.
C
C*****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON/PROBCV/UNIT,NSUB,NINDSB,NTOTAS,NINDAS,NGLC,IDOF,
     *NODE,IN,IO,NMBKSB,IDRY
C
      COMMON /LLCV/ ICLSUB,NJ,NLC,NLCBEL,NFIBN,NIBNS,NE
C
      DIMENSION X(1),Y(1),AREA(1),RI(1),YMOD(1),NOD(2,1),MKODE(1)
C
      WRITE(IO,2000)
50    READ(IN,1000) N,X(N),Y(N),INC
      IF(INC.EQ.0) GO TO 200
      NINT = (N-NOLD)/INC
      RN   = NINT
      IF(RN.LT.FLOAT(N-NOLD)/FLOAT(INC)-0.001) GO TO 999
      DX = (X(N) - X(NOLD))/RN
      DY = (Y(N) - Y(NOLD))/RN
      L = NOLD
      M = NINT - 1
      DO 100 J=1,M
      LL      = L + INC
      X(LL)   = X(L) + DX
      Y(LL)   = Y(L) + DY
      L = LL
100   CONTINUE
200   WRITE(IO,2100) N,X(N),Y(N),INC
      NOLD = N
      IF(N.LT.NJ) GO TO 50
C
      READ(IN,1100) ADEF,RDEF,YDEF
      WRITE(IO,2200) ADEF,RDEF,YDEF
C
      WRITE(IO,2300)
300   READ(IN,1200) M,(NOD(I,M),I=1,2),MKODE(M),INC,NOD1,NOD2,
     *AREA(M),RI(M),YMOD(M)
      IF(AREA(M).EQ.0.) AREA(M)  = ADEF
      IF(RI(M).EQ.0.)   RI(M)    = RDEF
      IF(YMOD(M).EQ.0.) YMOD(M)  = YDEF
      IF(INC.EQ.0) GO TO 500
      NINT = (M-MOLD)/INC - 1
      L    = MOLD
      DO 400 I=1,NINT
      LL        = L + INC
      AREA(LL)  = AREA(M)
      RI(LL)    = RI(M)
      YMOD(LL)  = YMOD(M)
      MKODE(LL) = MKODE(M)
      NOD(1,LL) = NOD(1,L) + NOD1
```

```
      NOD(2,LL) = NOD(2,L) + NOD2
400   L = LL
C
500   WRITE(IO,2400) M,NOD(1,M),NOD(2,M),MKODE(M),INC,NOD1,NOD2,
     *AREA(M),RI(M),YMOD(M)
      MOLD = M
      IF(M.LT.NE) GO TO 300
C
      READ(IN,1300) K
      IF(K.EQ.0) GO TO 600
      WRITE(IO,2700)
      WRITE(IO,2750) (N,X(N),Y(N),N=1,NJ)
      WRITE(IO,2800)
      WRITE(IO,2850) (M,(NOD(I,M),I=1,2),MKODE(M),AREA(M),RI(M),
     *YMOD(M),M=1,NE)
C
600   RETURN
C
999   WRITE(IO,9999) N
9999  FORMAT('NODAL GEOMETRY DATA INPUT ERROR',I5)
      STOP
C
C   FORMAT STATEMENTS
C
1000  FORMAT(I4,2F12.0,I4)
1100  FORMAT(3F12.0)
1200  FORMAT(7I4,3F12.0)
1300  FORMAT(I4)
2000  FORMAT(///'NODAL GEOMETRY DATA AS INPUT'//,4X,1HN,7X,1HX,
     *14X,1HY,9X,3HINC//)
2100  FORMAT(I5,2D15.6,I5)
2200  FORMAT(///'MEMBER PROPERTIES DEFAULT VALUES'//,
     *'AREA',10X,'=',D15.6,//,
     *'M.INERTIA',5X,'=',D15.6,//,
     *'Y.MODULUS',5X,'=',D15.6)
2300  FORMAT(///'MEMBER DATA AS INPUT'//,
     *4X,1HM,4X,1HI,4X,1HJ,1X,4HCODE,2X,3HINC,1X,4HINCI,1X,
     *4HINCJ,5X,4HAREA,10X,1HI,13X,4HYMOD//)
2400  FORMAT(7I5,3D15.6)
2700  FORMAT('1','COMPLETED NODAL GEOMETRY DATA'//,
     *4X,1HN,7X,1HX,14X,1HY//)
2750  FORMAT(I5,2D15.6)
2800  FORMAT('1','COMPLETED MEMBER DATA',//,
     *4X,1HM,4X,1HI,4X,1HJ,1X,4HCODE,5X,4HAREA,10X,1HI,13X,
     *4HYMOD//)
2850  FORMAT(4I5,3D15.6)
C
      END
```

```
            SUBROUTINE INPUT5
C
C THIS SEGMENT READS AN ASSEMBLY UNIT(ICLSUB) CONTROL VARIABLES.
C PROGRAM MUSAPP.
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      COMMON/PROBCV/UNIT,NSUB,NINDSB,NTOTAS,NINDAS,NGLC,IDOF,
     *NODE,IN,IO,NMBKSB,IDRY
C
      COMMON /HLCV1/ ICLSUB,NJH,NLCH,NEBELH,NFIBNH,NIBNSH
C
      COMMON /HLCV2/ NCU,MXND,IFLAG
C
      READ(IN,1000) ICH,NJH,NLCH,NEBELH,NFIBNH,NIBNSH,NCU,MXND,
     *IFLAG
      IF(ICH.NE.ICLSUB) GO TO 999
      WRITE(IO,2000) ICH,NJH,NLCH,NEBELH,NFIBNH,NIBNSH,NCU,MXND,
     *IFLAG
C
      RETURN
C
999   WRITE(IO,3000) ICLSUB,ICH
3000  FORMAT('ASSEMBLED UNIT(',I4,') IS NOT ENTERED IN ORDER.',
     *' ICH=',I4)
      STOP
C
C   FORMAT STATEMENTS
C
1000  FORMAT(9I4)
2000  FORMAT('1','INDEPENDENT UNIT NUMBER(',I4,
     *') CONTROL VARIABLES',/50(1H*)//,
     *'TOTAL NUMBER OF NODES                              =',I4//,
     *'TOTAL NUMBER OF LOCAL CASES OF LOADING             =',I4//,
     *'TOTAL NUMBER OF EXTERNAL BOUNDARY ELEMENTS         =',I4//,
     *'NUMBER OF FIRST INTER-BOUNDARY NODE                =',I4//,
     *'TOTAL NUMBER OF INTER-BOUNDARY NODES               =',I4//,
     *'TOTAL NUMBER OF CONSTITUENT UNITS                  =',I4//,
     *'MAXIMUM NUMBER OF INTER BOUNDARY NODES IN ANY OF'/,
     *'THE CONSTITUENT UNITS                              =',I4//,
     *' MASTER SYSTEM STORAGE FLAG                        =',I4,
     *'    WHERE, 0 = DO NOT STORE.',/,
     *58X,'AND    1 = STORE.')
C
      END
```

```
      SUBROUTINE INPUT6(ORINT,INDSUB,IRCSUB,IBNSUB,NPG,LCSUB)
C
C   THIS SUBROUTINE READS INDPENDENT ASSEMBLY UNIT(ICLSUB)
C   CONSTITUENT UNITS ORIENTATIONS AND THE CONNECTIVITY AND LOAD
C   CASES CONTROL ARRAYS. PROGRAMM MUSAPF.
C
C*******************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON/PROBCV/UNIT,NSUB,NINDSB,NTOTAS,NINDAS,NGLC,IDOF,
     *NODE,IN,IO,NMBKSB,IDRY
C
      COMMON /HLCV1/ ICLSUB,NJH,NLCH,NEBELH,NFIBNH,NIBNSH
C
      COMMON /HLCV2/ NCU,MXND,IFLAG
C
      DIMENSION ORINT(1),INDSUB(1),IRCSUB(1),IBNSUB(1),
     *NPG(MXND,1),LCSUB(NLCH,1)
C
      WRITE(IO,2000)
      NM = NCU/20
      NS1 = 1
      IF(NM.EQ.0) GO TO 20
C
      DO 10 KK=1,NM
      NS2 = KK*20
      READ(IN,1000)  (INDSUB(I),I=NS1,NS2)
      READ(IN,1000)  (IRCSUB(I),I=NS1,NS2)
      READ(IN,1000)  (IBNSUB(I),I=NS1,NS2)
      READ(IN,1000)  ((NPG(J,I),I=NS1,NS2),J=1,MXND)
      READ(IN,1000)  ((LCSUB(J,I),I=NS1,NS2),J=1,NLCH)
      WRITE(IO,2100)  (K,K=NS1,NS2)
      WRITE(IO,2200)  (INDSUB(I),I=NS1,NS2)
      WRITE(IO,2300)  (IRCSUB(I),I=NS1,NS2)
      WRITE(IO,2400)  (IBNSUB(I),I=NS1,NS2)
      WRITE(IO,2500)
      WRITE(IO,2600)  (J,(NPG(J,I),I=NS1,NS2),J=1,MXND)
      WRITE(IO,2700)
      WRITE(IO,2600)  (J,(LCSUB(J,I),I=NS1,NS2),J=1,NLCH)
      NS1 = NS2 + 1
10    CONTINUE
      IF(NS1.GT.NCU) GO TO 50
C
20    READ(IN,1000)  (INDSUB(I),I=NS1,NCU)
      READ(IN,1000)  (IRCSUB(I),I=NS1,NCU)
      READ(IN,1000)  (IBNSUB(I),I=NS1,NCU)
      WRITE(IO,2100)  (K,K=NS1,NCU)
      WRITE(IO,2200)  (INDSUB(I),I=NS1,NCU)
      WRITE(IO,2300)  (IRCSUB(I),I=NS1,NCU)
      WRITE(IO,2400)  (IBNSUB(I),I=NS1,NCU)
      WRITE(IO,2500)
      DO 30 J=1,MXND
      READ(IN,1000)  (NPG(J,I),I=NS1,NCU)
      WRITE(IO,2600)  (J,(NPG(J,I),I=NS1,NCU))
30     CONTINUE
C
      WRITE(IO,2700)
      DO 40 J=1,NLCH
      READ(IN,1000)  (LCSUB(J,I),I=NS1,NCU)
      WRITE(IO,2600)  (J,(LCSUB(J,I),I=NS1,NCU))
40    CONTINUE
C
50    READ(IN,1100)  (ORINT(I),I=1,NCU)
      WRITE(IO,2800)
      WRITE(IO,2900)  (I,ORINT(I),I=1,NCU)
C
      RETURN
C
C   FORMAT STATEMENTS
C
1000  FORMAT(20I4)
1100  FORMAT(8F10.0)
2000  FORMAT(///'CONSTITUENT UNITS INFORMATION ARRAYS'//)
2100  FORMAT('CONSTITUENT UNIT LOCAL NUMBER ',20I5)
2200  FORMAT('CONSTITUENT UNIT INDP. NUMBER ',20I5)
2300  FORMAT('RECURRENCE FLAG OF CONST. UNIT',20I5)
2400  FORMAT('NUMBER OF INTER-BOUNDARY NODES',20I5)
2500  FORMAT(//'CONSTITUENT UNITS CONNECTIVITY ARRAY'//,27X,
     *'LNP')
2600  FORMAT(25X,21I5)
2700  FORMAT(//'CONSTITUENT UNITS LOAD CASES CONTROL ARRAY'//,
     *28X,'LC')
2800  FORMAT(//'CONSTITUENT UNITS ORIENTATIONS (IN DEGREES)'//)
2900  FORMAT(8(I5,F10.4))
C
      END
```

C.3 The Data Managing Package

This package is a modification of a simple manager proposed by McCormick [8]. It is composed of functions ISPAC and LOCOM, subroutines REMOV, and REMOV2, and a BLOCK DATA segment which initializes the common block DIMCOM.

The function ISPAC is called from subroutine MAINMG to define an array in one of five common blocks. This function then calls function LOCOM to check if this array is currently defined in the same common block. If it is not, then ISPAC enters the name of the function in the name directory (NAMES) corresponding to the required common block. ISPAC then adds the length of this array to the current length of this common block, and checks if the updated length of the common block exceeds the maximum specified in MAIN. Finally it returns with the position of the first element of this array in the common block. Subroutine REMOV is called from MAINMG to remove any of the last defined arrays in any common block from the name directory (NAMES), and the pointer directory (IPT). Subroutine REMOV2 is called from MAINMG to free an entire common block by initializing the corresponding column in NAMES and IPT. The arguments of these segments are as follows

ISPAC (NAME, LEN, K)

LOCOM (NAME, K)

REMOV (NAME, K)

REMOV2 (K)

where, NAME is the name of an array to be defined, LEN is the array length, and K is a logical number to designate the required common block. The listing of this package follows.

```
      FUNCTION ISPAC(NAME,LENGTH,K)
C
C   A SIMPLE MANAGER WHICH WORKS WITH 5 FIXED LENGTH COMMON BLO-
C   CKS, A 5-COLUMN NAME DIRECTORY AND POINTER DIRECTORY.
C
C*****************************************************************
      REAL*8 NAMES,NAME
C
      CCMMON /DIMCOM/ LAST1,LAST2,LAST3,LAST4,LAST5,MAXDIM,
     *NAMES(5,20),IPT(5,21),ICOM(5)
C
C   CHECK IF NAME ALREADY EXISTS.
C
      ISPACE = LOCOM(NAME,K)
      IF(ISPACE.EQ.0) GO TO 10
      GO TO 100
C
C   ENTER NEW NAME IN DIRECTORY.
C
10    GO TO (20,30,40,50,60),K
20    LAST1 = LAST1 + 1
      LAST  = LAST1
      GO TO 70
30    LAST2 = LAST2 + 1
      LAST  = LAST2
      GO TO 70
40    LAST3 = LAST3 + 1
      LAST  = LAST3
      GO TO 70
50    LAST4 = LAST4 + 1
      LAST  = LAST4
      GO TO 70
60    LAST5 = LAST5 + 1
      LAST  = LAST5
C
70     IF(LAST.GT.MAXDIM) GO TO 200
      NAMES(K,LAST) = NAME
      ISPACE = IPT(K,LAST)
      IPT(K,LAST+1) = ISPACE + LENGTH
      IF((IPT(K,LAST+1)-1).GT.ICOM(K)) GO TO 300
      ISPAC = ISPACE
C
      RETURN
C
C   EXITS RESULTING FROM DIAGNOSED ERRORS
C
100   WRITE(6,1000) NAME
1000  FORMAT(22H***NAME ALREADY EXISTS,10X,A8)
      GO TO 400
200   WRITE(6,2000) NAME,K
2000  FORMAT(17H***TABLE OVERFLOW ,10X,A8,I4)
      GO TO 400
300   WRITE(6,3000) NAME,K,IPT(K,LAST),LENGTH
3000  FORMAT(23H***COMMON AREA OVERFLOW ,A8,3I4)
400   CALL EXIT
C
      END
```

```
      FUNCTICN LCCOM(NAME,K)
C
C   LOCATES INDEX OF A GIVEN NAME IN NAMES DIRECTORY.
C
C*****************************************************************
      REAL*8 NAME,NAMES
C
      COMMON /DIMCOM/ LAST1,LAST2,LAST3,LAST4,LAST5,MAXDIM,
     *NAMES(5,20),IPT(5,21),ICOM(5)
C
      GO TO (10,20,30,40,50),K
10    LAST = LAST1
      GO TO 60
20    LAST = LAST2
      GO TO 60
30    LAST = LAST3
      GO TO 60
40    LAST = LAST4
      GO TO 60
50    LAST = LAST5
C
60    IF(LAST.EQ.0) GO TO 200
      DO 100 M=1,LAST
      IF(NAMES(K,M).NE.NAME) GO TO 100
      LOCOM = M
      RETURN
100   CONTINUE
200   LOCOM = 0
C
      RETURN
C
      END
```

```
      SUBROUTINE REMOV(NAME,K)
C
C   REMOVES NAME, IF IT IS THE LAST VARIABLE IN COLUMN K, IN
C   DIRECTORY, AND UPDATES POINTERS ACCORDINGLY.
C
C*****************************************************************
      REAL*8 NAME,NAMES
C
      COMMON /DIMCOM/ LAST1,LAST2,LAST3,LAST4,LAST5,MAXDIM,
     *NAMES(5,20),IPT(5,21),ICOM(5)
C
      GO TO (10,20,30,40,50),K
10    LAST = LAST1
      GO TO 60
20    LAST = LAST2
      GO TO 60
30    LAST = LAST3
      GO TO 60
40    LAST = LAST4
      GO TO 60
50    LAST = LAST5
C
60    IF(NAMES(K,LAST).NE.NAME)   GO TO 150
C
C   LAST VARIABLE IN DIRECTORY COLUMN K IS NAME;  REMOVE IT.
C
      IPT(K,LAST+1)  = 0
      NAMES(K,LAST)  = 0
      GO TO (70,80,90,100,110),K
70    LAST1 = LAST1 - 1
      GO TO 120
80    LAST2 = LAST2 - 1
      GO TO 120
90    LAST3 = LAST3 - 1
      GO TO 120
100   LAST4 = LAST4 - 1
      GO TO 120
110   LAST5 = LAST5 - 1
C
120   RETURN
C
150   WRITE(6,1500) NAME,NAMES(K,LAST)
1500  FORMAT(37H***NAME IS NOT LAST VARIABLE IN NAMES  ,2A8)
      CALL EXIT
C
      END
```

```
      SUBROUTINE REMOV2(K)
C
C   INITIALISES COLUMN K IN NAMES AND IPT, AND LASTK
C
C*****************************************************************
      REAL*8 NAME,NAMES
C
      COMMON /DIMCOM/ LAST1,LAST2,LAST3,LAST4,LAST5,MAXDIM,
     *NAMES(5,20),IPT(5,21),ICOM(5)
C
      GO TO (10,20,30,40,50),K
10    LAST = LAST1
      LAST1=0
      GO TO 60
20    LAST = LAST2
      LAST2=0
      GO TO 60
30    LAST = LAST3
      LAST3=0
      GO TO 60
40    LAST = LAST4
      LAST4=0
      GO TO 60
50    LAST = LAST5
      LAST5=0
C
60    LASTN = LAST + 1
      DO 100 J=2,LASTN
      J1 = J - 1
      NAMES(K,J1)  = 0
      IPT(K,J)     = 0
100   CONTINUE
C
      RETURN
C
      END
```

```
      BLOCK DATA
C
      REAL*8 NAME,NAMES
C
      COMMON /DIMCOM/ LAST1,LAST2,LAST3,LAST4,LAST5,MAXDIM,
     *NAMES(5,20),IPT(5,21),ICOM(5)
C
      DATA LAST1,LAST2,LAST3,LAST4,LAST5,MAXDIM,IPT(1,1),IPT(2,1
     *),IPT(3,1),IPT(4,1),IPT(5,1)/5*0,20,5*1/
C
      END
```

C.4 Data Storage and Retrieval Package

This package is composed of six subroutines. Subroutines CLEAR, and ICLEAR initialize real and integer arrays, respectively. Subroutines RTRV1 and RTRV2 retrieve respectively integer and real arrays from backing storage, while subroutines STORE1, and STORE2, store respectively integer and real array in backing storage. The arguments of these subroutines are as follows

```
CLEAR (ARRAY, LEN)
ICLEAR (IARRAY, LEN)
RTRV1 (IN, IARRAY, LEN)
RTRV2 (IN, ARRAY, LEN)
STORE1 (IO, IARRAY, LEN)
STORE2 (IO, ARRAY, LEN)
```

where, ARRAY is the first element of a real array to be initialized, retrieved, or stored, IARRAY is the first element of an integer array to be initialized, retrieved, or stored, LEN is the length of an array, IN,IO is a logical number which designates a sequential file. The listing of this package follows.

```
      SUBROUTINE CLEAR(ARRAY,LEN)
C
C   THIS SUBROUTINE ASSIGNES 0.0 TO A REAL ARRAY OF LENGTH LEN.
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION ARRAY(1)
C
      DO 100 I=1,LEN
      ARRAY(I) = 0.0
100   CONTINUE
C
      RETURN
C
      END
```

```
      SUBROUTINE RTRV1(IN,IARRAY,LEN)
C
C   RETRIEVES AN INTEGER ARRAY OF LENGTH LEN, FROM FILE(IN).
C
C*****************************************************************
C
      DIMENSION IARRAY(LEN)
C
      READ(IN) IARRAY
C
      RETURN
C
      END
```

```
      SUBROUTINE ICLEAR(IARRAY,LENGTH)
C
C   THIS SEGMENT INITIALISES AN INTEGER ARRAY OF LENGTH(LENGTH).
C
C*****************************************************************
C
      DIMENSION IARRAY(1)
C
      DO 100 I=1,LENGTH
      IARRAY(I) = 0
100   CONTINUE
C
      RETURN
C
      END
```

```
      SUBROUTINE RTRV2(IN,ARRAY,LEN)
C
C   RETRIEVES A REAL ARRAY OF LENGTH LEN, FROM FILE (IN).
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION ARRAY(LEN)
C
      READ(IN)ARRAY
C
      RETURN
C
      END
```

```
      SUBROUTINE STORE1(IO,IARRAY,LEN)
C
C    STORES AN INTEGER ARRAY OF LENGTH LEN ON FILE(IO).
C
C**********************************************************************
C
      DIMENSION IARRAY(LEN)
C
      WRITE(IO) IARRAY
C
      RETURN
C
      END
```

```
      SUBROUTINE STORE2(IO,ARRAY,LEN)
C
C    STORES A REAL ARRAY OF LENGTH LEN ON FILE(IO)
C
C**********************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION ARRAY(LEN)
C
      WRITE(IO) ARRAY
C
      RETURN
C
      END
```

C.5 The Equation Solving Package

C.5.1 Description of Package

This package is based on the algorithms derived in Chapter 2 for substructured columnwise decomposition and backsubstitution. It is composed of nine segments for which the description and listing follows. The arguments are described in Section C.5.3.

C.5.2 Description of Subroutines

(1) ADDRES (MAXA, MHT, NN, ID, NWA)

This is a simplified version of subroutine ADRESS [2]. It computes the addresses of the diagonal components of a stiffness matrix stored columnwise in a vector, as well as the total length of this vector.

(2) COLHT (NB, NODES, ND, ID, ME, MHT, NP, LM)

This is a modified version of subroutine COLHT [2]. It forms and updates the active column heights of a stiffness matrix, see Section 2.7.1. It is called per structural element which forms a part of a basic substructure unit, or per substructure unit which forms a part of a higher level assemblage.

(3) EQSBST (A, B, MAXA, NP, ID, NLC)

This subroutine is a modified version of subroutine COLSOL [2]. It performs a columnwise decomposition of the internal degrees of freedom of a substructure stiffness matrix and reduces the right hand side up to the same level. It is based on the algorithm shown in Fig. 2.6 and can handle any number of cases of loading.

(4) EQFT (A, MAXA, NP, ID, NN)

This subroutine forms the partition $[F]^T$ of a substructure stiffness matrix, and is based on the algorithm shown in Fig. 2.7.

(5) EQKBB (A, B, MAXA, NP, ID, NN, NLC)

This subroutine forms the inter-boundary stiffness and load partitions $[K]^*$ and $\{R_b^*\}$, and is based on the algorithm shown in Fig. 2.9.

(6) BSKB1 (B, A, MAXA, NP, NLC, ID, KB)

This subroutine performs the backsubstitution process through the internal degrees of freedom, and is based on the lower algorithm of Fig. 2.9.

(7) BKSK2 (A, B, MAXA, NPLI, NJ, MNLC, ID)

This subroutine performs the backsubstitution upper algorithm of Fig. 2.9.

(8) MODAX1 (BB, MAXB, MHB, NN, ID, NWK)

This subroutine checks the columns of the assembled stiffness matrix of a higher level unit stored in array BB for the actual first non zero component, changes array MAXB accordingly, and shifts the components of BB forward as may be necessary.

(9) SKYPRD (MAXA, NPG, MHB, ID, NFIBN, NIBNS, MXND, ME)

This subroutine forms the column heights array MHB for a higher level unit stiffness matrix taking into consideration the sky-lines of the lower level units.

C.5.2 Description of the Arguments

A : The stiffness storage vector.

B : The load array.

BB	:	A higher level unit stiffness storage vector.
ID	:	Number of degrees of freedom per node.
KB	:	0, if the problem is unsubstructured, and 1, if otherwise.
LM	:	The degrees of freedom associated with an element or a lower level unit storage vector.
MAXA	:	The diagonal component addressing array.
MAXB	:	The diagonal component addressing array of a higher level unit.
ME	:	Element number, or the local constituent unit number.
MHB	:	The column height array of a higher level unit.
MHT	:	The column height array.
MXND	:	The maximum number of inter-boundary nodes in any constituent unit.
ND	:	Number of degrees of freedom associated with an element, or with the inter-boundary partition of a constituent unit.
NFIBN	:	Number of the first inter-boundary node in a constituent unit.
NIBNS	:	Number of inter-boundary nodes in a constituent unit.
NLC	:	Number of cases of loading.
NN	:	Total number of nodes in a unit.
NP	:	Total number of internal nodes in a unit.

```
      SUBROUTINE ADDRES(MAXA,MHT,NN,ID,NWA)
C
C   THIS SUBROUTINE CALCULATES THE ADDRESSES OF THE DIAGONAL
C   ELEMENTS  AND LENGTH OF A STIFFNESS MATRIX UPPER TRIANGLE
C   STORED COLUMN-WISE UNDER A SKYLINE.
C
C**************************************************************
C
      DIMENSION MAXA(1),MHT(1)
C
      NEQ = NN*ID
      NM  = NEQ + 1
C
      MAXA(1) = 1
C
      IF(NEQ.EQ.1) GO TO 30
      DO 20 I=1,NEQ
      MAXA(I+1) = MAXA(I) + MHT(I) + 1
 20   CONTINUE
C
      NWA = MAXA(NM) - 1
      GO TO 40
C
 30   NWA = 1
 40   RETURN
C
      END
```

```
      SUBROUTINE COLHT(NB,NODES,ND,ID,ME,MHT,NP,LM)
C
C   THIS SUBROUTINE IS CALLED PER ELEMENT, OR PER SUBSTRUCTURE
C   TO FORM AND UPDATE THE COLUMN HEIGHT ARRAY(MHT).
C
C******************************************************************
C
      DIMENSION MHT(1),NP(NB,1),LM(1)
C
      DO 100 I=1,NODES
      II = I*ID + 1
      N  = NP(I,ME)*ID + 1
      DO 100 J=1,ID
      JJ = II - J
100   LM(JJ) = N - J
C
      LS = 10000
C
      DO 200 I=1,ND
      IF(LM(I)-LS) 150,200,200
150   LS = LM(I)
200   CONTINUE
C
      DO 300 I=1,ND
      II = LM(I)
      MB = II - LS
      IF(MB.GT.MHT(II)) MHT(II) = MB
300   CONTINUE
C
      RETURN
C
      END
```

```
      SUBROUTINE  EQSBST(A,B,MAXA,NP,ID,NLC)
C
C   THIS SUBROUTINE TRIANGULARIZES A STIFFNESS MATRIX STORED
C   COLUMNWISE UNDER A SKYLINE AND REDUCES THE CORRESPONDING
C   LOAD VECTOR, DOWN TO EQUATION NP*ID.
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION A(1),B(NLC,1),MAXA(1)
C
      NB = NP*ID
C
      DO  1000 N=1,NB
      KN = MAXA(N)
      KL = KN + 1
      KU = MAXA(N+1) - 1
      KH = KU - KL
      IF(KH) 900,500,100
100   K  = N - KH
      IC = 0
      KLT= KU
      DO 400 J=1,KH
      IC = IC + 1
      KLT= KLT - 1
      KI = MAXA(K)
      ND = MAXA(K+1) - KI - 1
      IF(ND) 400,400,200
200   KK = IC
      IF(KK.GT.ND) KK = ND
      C  = 0.0
      DO 300 L = 1,KK
300   C  = C + A(KI+L)*A(KLT+L)
      A(KLT) = A(KLT) - C
400   K  = K + 1
500   K  = N
C
      C  = 0.0
      DO 600 KK = KL,KU
      K  = K - 1
      KI = MAXA(K)
      D  = A(KK)/A(KI)
      C  = C + D*A(KK)
      A(KK) = D
600   CONTINUE
      A(KN) = A(KN) - C
C
      DO 800 IC=1,NLC
      K  = N
      C = 0.0
      DO 700 KK = KL,KU
      K  = K - 1
      C  = C + A(KK)*B(IC,K)
700   CONTINUE
      B(IC,N) = B(IC,N) - C
800   CONTINUE
C
900   IF(A(KN)) 950,950,1000
950   WRITE(6,3000) N,A(KN)
      STOP
C
1000  CONTINUE
C
      RETURN
C
3000  FORMAT('ZERO OR NEGATIVE ELEMENT ON MAIN DIAGONAL NO.',I4,
     *D15.6)
C
      END
```

```
      SUBROUTINE EQPT(A,MAXA,NP,ID,NN)
C
C   THIS SUBROUTINE IS A PART OF THE PARTIAL REDUCTION PACKAGE.
C   IT FORMS  F(T), FOR AN UPPER TRINGULAR MATRIX STORED COLUM-
C   NWISE, UNDER A SKYLINE.
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION A(1),MAXA(1)
C
      NS = NP*ID + 1
      NB = NN*ID
      NI = NP*ID
C
      DO 700 N=NS,NB
      KN = MAXA(N)
      KL = KN + N - NI
      KU = MAXA(N+1) - 1
      KH = KU - KL
      IF (KH) 700,500,100
100   K  = N - (KU-KN) + 1
      IC = 0
      KLT= KU
      DO 400 J=1,KH
      IC = IC + 1
      KLT= KLT - 1
      KI = MAXA(K)
      ND = MAXA(K+1) - KI - 1
      IF(ND)400,400,200
200   KK = IC
      IF(KK.GT.ND) KK = ND
      C  = 0.0
      DO 300 L=1,KK
300   C  = C + A(KI+L)*A(KLT+L)
      A(KLT) = A(KLT) - C
400   K  = K + 1
500   K  = NS
C
      DO 600 KK=KL,KU
      K  = K - 1
      KI = MAXA(K)
      A(KK) = A(KK)/A(KI)
600   CONTINUE
700   CONTINUE
C
      RETURN
C
      END
```

```
      SUBROUTINE EQKBB(A,B,MAXA,NP,ID,NN,NLC)
C
C   THIS SUBROUTINE IS A PART OF THE PARTIAL REDUCTION PACKAGE.
C   IT FORMS THE  PARTITION KBB*, AND RBB*(THE INTER-BOUNDARY SUB-
C   ARRAYS).
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(1),B(NLC,1),MAXA(1)
C
      NS = NP*ID + 1
      NB = NN*ID
      NI = NP*ID
C
      DO 600 N=NS,NB
      KN  = MAXA(N)
      KU1 = KN + N - NS
      KU2 = MAXA(N+1) - 1
      IF(KU1.GE.KU2) GO TO 600
      KH1 = KU2 - KU1
      KH2 = KU1 - KN + 1
      K   = N - KH2 + 1
      KLT = KU1 + 1
C
      DO 300 J = 1,KH2
      KLT = KLT - 1
      KI  = MAXA(K)
      KI1 = KI + K - NS
      KI2 = MAXA(K+1) - 1
      KHI = KI2 - KI1
      IF(KHI) 300,300,100
100   KK  = KHI
      IF(KK.GT.KH1) KK = KH1
      C  = 0.0
      L1 = KI + J + KK
      L2 = KLT + J + KK
      L3 = NI - KK
      DO 200 L=1,KK
200   C  = C + A(L1-L)*A(L2-L)*A(MAXA(L3+L))
      A(KLT) = A(KLT) - C
300   K  = K + 1
C
      L4 = NI - KH1
      L5 = KU2 + 1
      DO 500 IC=1,NLC
      C = 0.0
      DO 400 L=1,KH1
400   C = C + A(L5-L)*B(IC,(L4+L))
      B(IC,N) = B(IC,N) - C
500   CONTINUE
600   CONTINUE
C
      RETURN
C
      END
```

```
      SUBROUTINE BKSB1(B,A,MAXA,NJ,NLC,ID,KB)
C
C     THIS SUBROUTINE IS A PART OF THE PARTIAL REDUCTION PACKA-
C     GE. IT FORMS THE LAST STAGE OF BACKSUBSTITUTION,THE OPER-
C     ATION (L(-1)(T)*RI*). IN CASE OF AN UNSUBSTRUCTURED PROB-
C     LEM IT PERFORMS THE OPERATION //D(-1)*L(-1)(T)*R//
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION A(1),B(NLC,1),MAXA(1)
C
      IF(KB.NE.0) GO TO 200
      NN = ID*NJ
      DO 100 I=1,NLC
      DO 100 J=1,NN
      K = MAXA(J)
100   B(I,J) = B(I,J)/A(K)
C
200   NN = ID*NJ
      N  = NN
      DO 600 L=2,NN
      KL = MAXA(N) + 1
      KU = MAXA(N+1) - 1
      IF(KU-KL) 600,300,300
300   DO 500 I=1,NLC
      IF(B(I,N).EQ.0.) GO TO 500
      K  = N
      DO 400 KK=KL,KU
      K = K - 1
400   B(I,K) = B(I,K) - A(KK)*B(I,N)
500   CONTINUE
600   N  = N - 1
C
      RETURN
C
      END
```

```
      SUBROUTINE BKSB2(A,BA,MAXA,NPLI,NJ,MNLC,ID)
C
C     THIS SUBROUTINE IS A PART OF THE PARTIAL REDUCTION PACKA-
C     GE. IT FORMS A PART OF THE BACKSUBSTITUTION SCHEME,NAMELY
C     THE QUANTITY // D(-1)*RI*- F(T)*RBB//.
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION A(1),BA(MNLC,1),MAXA(1)
C
      NS = (NPLI-1)*ID + 1
      NN = ID*NJ
      NI = NS - 1
C
      DO 100 I=1,MNLC
      DO 100 N=1,NI
      KN = MAXA(N)
      BA(I,N) = BA(I,N)/A(KN)
100   CONTINUE
C
      DO 400 N=NS,NN
      KN  = MAXA(N)
      KU1 = KN + N - NI
      M1  = MAXA(N+1)
      KU2 = M1 - 1
      IF(KU1.GT.KU2) GO TO 400
      KH  = KU2 - KU1 + 1
      K   = N - KU2 + KN
C
      DO 300 J=1,KH
      AR = A(M1 - J)
      IF(AR.EQ.0.) GO TO 300
      DO 200 I=1,MNLC
200   BA(I,K) = BA(I,K) - BA(I,N)*AR
300   K = K + 1
C
400   CONTINUE
C
      RETURN
C
      END
```

```
      SUBROUTINE MODAX1(BB,MAXB,MHB,NJH,IDOF,NWK)
C
C     THIS SEGMENT MODIFIES THE SKYLINE OF AN ASSEMBLED
C     HIGHER LEVEL UNIT STIFFNESS MATRIX.
C
C*******************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION BB(1),MAXB(1),MHB(1)
C
      NN = IDOF*NJH
      MHB(NN+1) = 0
      IS1 = 0
      IS2 = 0
      DO 400 I=1,NN
      KU = MAXB(I+1) - IS1
      KH = MHB(I) + 1
      DO 100 J=1,KH
      IF(BB(KU-J).EQ.0.0) GO TO 100
      GO TO 200
100   CONTINUE
200   IS2 = IS2 + J - 1
      IF(IS2.EQ.0) GO TO 400
      KH1 = MHB(I+1) + 1
      KL = KU + IS1 - 1
      JJ = KL - IS2
      DO 300 KK=1,KH1
      BB(JJ+KK) = BB(KL+KK)
300   CONTINUE
      MAXB(I+1) = JJ + 1
      IS1 = IS2
400   CONTINUE
      NWK = MAXB(NN+1) - 1
C
      RETURN
      END
```

```
      SUBROUTINE SKYPRD(MAXA,NPG,MHB,IDOF,NFIBN,NIBNS,MXND,IG)
C
C     THIS SEGMENT FORMS THE SKYLINE OF A HIGHER LEVEL UNIT
C     TAKING INTO CONSIDERATION THE SKYLINES OF THE CONSTI-
C     TUENT UNITS.
C
C*******************************************************************
      DIMENSION MAXA(1),MHB(1),NPG(MXND,1)
C
      NPP = NFIBN
      DO 500 M=1,NIBNS
      NGG  = NPG(M,IG)
      NGGG = IDOF*NGG
      NPPP = IDOF*NPP
      NPL  = MAXA(NPPP)
      NPM  = MAXA(NPPP-1)
      MAX  = (NPL-NPM+1)/IDOF
      IF(MAX.GT.M) MAX = M
C
      DO 400 II=1,MAX
      NG = NPG((M-II+1),IG)
      IF(NG.GT.NGG) GO TO 200
      NH = NGGG - IDOF*NG +IDOF - 1
      IF(MHB(NGGG).GE.NH) GO TO 400
      DO 100 J=1,IDOF
100   MHB(NGGG-J+1) = NH - J + 1
      GO TO 400
C
200   NG = IDOF*NG
      NH = NG - NGGG + IDOF - 1
      IF(MHB(NG).GE.NH) GO TO 400
      DO 300 J=1,IDOF
300   MHB(NG-J+1) = NH - J + 1
C
400   CONTINUE
C
      NPP = NPP + 1
500   CONTINUE
C
      RETURN
      END
```

C.6 The Formulation and Output Package

This group of subroutines performs various tasks, and all are common to SISAPF and MUSAPF, except LOADID which is used only in MUSAPF. Subroutines MLOADS, DISPL, and STRESS are adapted from Reference 10. Descriptions of the subroutines can be found in the listing which follows. These subroutines are in alphabetical order.

ASSEMB

BOUND

BOUND2

DISPL

JLOAD

LOADID

MLOADS

RESUB

STIFF

STRESS

```
      SUBROUTINE ASSEMB(NPG,MAXB,LCSUB,MAXA,CRINT,BP,EB,B,A,IG,I
     *,NFIBN,NIBNS,MXND,NLC,NLCH)
C
C    THIS SUBROUTINE ASSEMBLES THE KEB*,RBB* PARTITIONS OF A LOW-
C    ER LEVEL UNIT INTO THIER CORRESPONDING LOCATIONS IN HIGHER
C    LEVEL ARRAYS BB, BP, WITH THE APPROPRIATE TRANSFORMATIONS.
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION NPG(MXND,1),MAXB(1),LCSUB(NLCH,1),MAXA(1),CRINT(
     *1),BP(NLCH,1),EB(1),B(NLC,1),A(1)
C
      NEP   = NFIBN
      THETA = CRINT(IG)
      IF(THETA.EQ.0.) GO TO 10
      THETA = THETA*3.14159265/180.
      S  = DSIN(THETA)
      C  = DCOS(THETA)
      S2 = S**2
      C2 = C**2
      CS = C*S
C
10    DO 600 M=1,NIBNS
      NGG = NPG(M,IG)
      NGGG= 3*NGG
      NEPP= NEP*3
      NP3 = MAXA(NPPP)
      NP2 = MAXA(NPPP-1)
      NP1 = MAXA(NEPP-2)
      NG3 = MAXB(NGGG)
      NG2 = MAXB(NGGG-1)
      NG1 = MAXB(NGGG-2)
      NG12 = NG2 + 1
      NG13 = NG3 + 2
      NG23 = NG3 + 1
C
      T11 = A(NP1)
      T22 = A(NP2)
      T33 = A(NP3)
      T12 = A(NP2+1)
      T13 = A(NP3+2)
      T23 = A(NP3+1)
C
      IF(THETA.EQ.0.) GO TO 20
      X11 = C2*T11 + S2*T22 - 2.*CS*T12
      X12 = CS*(T11-T22) + (C2-S2)*T12
      X13 = C*T13 - S*T23
      X22 = C2*T22 + S2*T11 + 2.*CS*T12
      X23 = S*T13 + C*T23
      GO TO 30
C
20    BB(NG1)  = BB(NG1)   + T11
      BB(NG2)  = BB(NG2)   + T22
      BB(NG3)  = EB(NG3)   + T33
      BB(NG12)= BB(NG12)  + T12
      BB(NG13)= EB(NG13)  + T13
      BB(NG23)= BB(NG23)  + T23
C
      GO TO 40
30    BB(NG1)  = EB(NG1)   + X11
      BB(NG2)  = BB(NG2)   + X22
      BB(NG3)  = BB(NG3)   + T33
      BB(NG12)= BB(NG12)  + X12
      BB(NG13)= EB(NG13)  + X13
      BB(NG23)= BB(NG23)  + X23
C
40    MAX = (NP2 - NP1 - 1)/3
      IF(MAX.GT.(M-1)) MAX = M - 1
      IF(MAX.LE.0) GO TO 500
C
      DO 400 II=1,MAX
      I3  = 3*II
      T11 = A(NP1+I3)
      T12 = A(NP2+I3+1)
      T13 = A(NP3+I3+2)
      T21 = A(NP1+I3-1)
      T22 = A(NP2+I3)
      T23 = A(NP3+I3+1)
      T31 = A(NP1+I3-2)
      T32 = A(NP2+I3-1)
      T33 = A(NP3+I3)
      IF(THETA.EQ.0.) GO TO 100
C
      X11 = C2*T11 + S2*T22 - CS*(T12+T21)
      X22 = C2*T22 + S2*T11 + CS*(T12+T21)
      X21 = CS*(T11-T22) + C2*T21 - S2*T12
      X12 = CS*(T11-T22) + C2*T12 - S2*T21
      X31 = C*T31 - S*T32
      X32 = S*T31 + C*T32
      X13 = C*T13 - S*T23
      X23 = S*T13 + C*T23
C
100   NG = NPG((M-II),IG)
      IF(NG.GT.NGG) GO TO 200
C
      NH = NGGG - 3*NG
      N11 = NG1 + NH
      N21 = N11 - 1
      N31 = N11 - 2
      N22 = NG2 + NH
      N12 = N22 + 1
      N32 = N22 - 1
      N33 = NG3 + NH
      N23 = N33 + 1
      N13 = N33 + 2
C
      GO TO 300
C
200   NG  = 3*NG
```

```
      NH  = NG - NGGG
      N11 = MAXB(NG-2)  + NH
      N22 = MAXB(NG-1)  + NH
      N33 = MAXB(NG)  + NH
      N21 = N22 + 1
      N31 = N33 + 2
      N12 = N11 - 1
      N32 = N33 + 1
      N13 = N11 - 2
      N23 = N22 - 1
C
300   IF(THETA.EQ.0.) GO TO 350
      BB(N11)  = BB(N11)  + X11
      BB(N21)  = BB(N21)  + X21
      BB(N31)  = BB(N31)  + X31
      BB(N12)  = BB(N12)  + X12
      BB(N22)  = BB(N22)  + X22
      BB(N32)  = BB(N32)  + X32
      BB(N13)  = BB(N13)  + X13
      BB(N23)  = BB(N23)  + X23
      BB(N33)  = BB(N33)  + T33
C
      GO TO 400
350   BB(N11)  = BB(N11)  + T11
      BB(N21)  = BB(N21)  + T21
      BB(N31)  = BB(N31)  + T31
      BB(N12)  = BB(N12)  + T12
      BB(N22)  = BB(N22)  + T22
      BB(N32)  = BB(N32)  + T32
      BB(N13)  = BB(N13)  + T13
      BB(N23)  = BB(N23)  + T23
      BB(N33)  = BB(N33)  + T33
C
400   CONTINUE
C
500   DO 550 J=1,NLCH
      IK = LCSUB(J,I)
      IF(THETA.EQ.0.) GO TO 525
      BP(J,NGGG-2)  = BP(J,NGGG-2) +C*B(IK,NPPP-2) -S*B(IK,NPPP-1)
      BP(J,NGGG-1)  = BP(J,NGGG-1) +S*B(IK,NPPP-2) +C*B(IK,NPPP-1)
      BP(J,NGGG)    = BP(J,NGGG)   +  B(IK,NPPP)
      GO TO 550
C
525   BP(J,NGGG-2)  = BP(J,NGGG-2) +B(IK,NPPP-2)
      BP(J,NGGG-1)  = BP(J,NGGG-1) +B(IK,NPPP-1)
      BP(J,NGGG)    = BP(J,NGGG)   +B(IK,NPPP)
550   CONTINUE
C
      NPP = NPP + 1
600   CONTINUE
C
      RETURN
C
      END
```

```
         SUBROUTINE BOUND(BES,NPB,KCDE,NEBEL,IN,IO)
C
C   THIS SEGMENT  READS THE EXTERNAL BOUNDARY CONDITIONS
C   PROGRAMM MULT.LEV.SUBSTAPF.
C
C*******************************************************************
         IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION BES(1),NPB(1),KODE(1)
C
      DO 100 J=1,NEBEL
      READ(IN,1000) N,NPB(N),KODE(N),BES(N)
100   IF(BES(N).EQ.0.) BES(N) = 1.0E20
      WRITE(IO,2000)
      WRITE(IO,2100)  (N,NPB(N),KODE(N),BES(N),N=1,NEBEL)
C
       RETURN
C
C   FORMAT STATEMENTS
C
1000  FORMAT(3I4,F12.0)
2000  FORMAT('1',T30,'EXTERNAL BOUNDARY ELEMENTS DATA'//,
     *3(4X,1HN,2X,3HNPB,1X,4HCODE,10X,2HEB,3X)//)
2100  FORMAT(3(3I5,D15.6))
C
      END
```

```
      SUBROUTINE BOUND2(A,MAXA,NPB,KODE,BES,NEBEL)
C
C     ADDS EXTERNAL BOUNDARY CONDITIONS TO STIFFNESS MATRIX A
C
C*********************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION A(1),MAXA(1),NPB(1),KODE(1),BES(1)
C
      DO 300 K = 1,NEBEL
      N      = NPB(K)
      KOD    = KODE(K)
      EX     = BES(K)
      NN     = 3*N
      IF((KOD-100).LT.0) GO TO 100
      NK     = MAXA(NN-2)
      A(NK)  = A(NK) + EX
      KOD    = KOD - 100
100   IF((KOD-10).LT.0) GO TO 200
      NK     = MAXA(NN-1)
      A(NK)  = A(NK) + EX
      KOD    = KOD -10
200   IF(KOD.EQ.0) GO TO 300
      NK     = MAXA(NN)
      A(NK)  = A(NK) + EX
300   CONTINUE
C
      RETURN
C
      END
```

```
      SUBROUTINE DISPL(FEF2,B,FEF,KSUB,K,I,IG,NJ,NE,NLC,NGLC,
     *IN,IO,NTOTAS)
C
C   DISPL OUTPUTS NODAL DISPLACEMENTS FOR BASIC UNIT(IG), AND
C   LOAD CASE(K). IT PREPARES DUMMY VECTOR FEF2 FOR COMPUTATION
C   OF MEMBER END FORCES.
C
C*********************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION FEF2(1),B(NGLC,1),FEF(NLC,1),KSUB(NGLC,1)
C
      IF(K.GT.1) GO TO 50
      WRITE(IO,2200) IG,K
      GO TO 75
C
50    WRITE(IO,2000) K
75    DO 100 N=1,NJ
      N3 = N*3
      N1 = N3 - 2
100   WRITE(IO,2100) N,(B(K,NN),NN=N1,N3)
C
      LC = K
      IF(NTOTAS.EQ.1) GO TO 200
      LC = KSUB(K,I)
200   DO 300 II=1,NE
      III = 6*II + 1
      DO 300 JJ=1,6
      FEF2(III-JJ) = FEF(LC,III-JJ)
300   CONTINUE
C
      RETURN
C
C   FORMAT STATEMENTS
C
2000  FORMAT('1','LOAD CASE NO. ',I4,//
     *,'NODAL DISPLACEMENTS',//
     *,4X,'N',7X,'U',14X,'V',14X,'R'//)
2100  FORMAT(I4,3D15.6)
2200  FORMAT('1',T30,'SUBSTRUCTURE NUMBER',I4,//29X,23(1H*),
     *//,'LOAD CASE NUMBER',I4,//,'NODAL DISPLACEMENTS'//
     *,4X,'N',7X,'U',14X,'V',14X,'R'//)
C
      END
```

```
      SUBROUTINE JLOAD(JKODE,B,BES,NLC,IN,IO)
C
C   THIS SEGMENT READS JOINT LOADS AND/OR DISPLACEMENTS FOR
C   INDEPENDENT UNIT(ICLSUB).
C
C*********************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION JKODE(1),B(NLC,1),NBEL(3),BES(1)
C
      DO 100 I=1,NLC
100   JKODE(I) = 0
C
      WRITE(IO,2000)
200   READ(IN,1000) N,NMLC,IDESP,(NBEL(I),I=1,3),INC
      IF(N.LE.0)  GO TO 700
      N3 = 3*N
      N2 = N3 - 1
      N1 = N3 - 2
      READ(IN,1100)  (LC,JKODE(LC),(B(LC,NN),NN=N1,N3),K=1,NMLC)
      IF(IDESP.EQ.0)  GO TO 300
C
      DO 275 J=1,NLC
      K = JKODE(J)
      IF((K-100).LT.0)  GO TO 225
      B(J,N1)  = B(J,N1)*BES(NBEL(1))
      K = K - 100
225   IF((K-10).LT.0)  GO TO 250
      B(J,N2)  = B(J,N2)*BES(NBEL(2))
      K = K - 10
250   IF(K.EQ.0) GO TO 275
      B(J,N3)  = B(J,N3)*BES(NBEL(3))
275   CONTINUE
C
300   IF(INC.EQ.0)  GO TO 500
      NINT = (N-NOLD)/INC - 1
      L = NOLD
      DO 400 I=1,NINT
      LL = L + INC
      L3 = LL*3
      L2 = L3 - 1
      L1 = L3 - 2
      DO 350 J=1,NLC
      B(J,L1)  = B(J,N1)
      B(J,L2)  = B(J,N2)
      B(J,L3)  = B(J,N3)
350   CONTINUE
400   L = LL
C
500   WRITE(IO,2100)  N,NMLC,(NBEL(I),I=1,3),INC
      WRITE(IO,2200)  (LC,JKODE(LC),(B(LC,NN),NN=N1,N3),LC=1,NLC)
      NOLD = N
      GO TO 200
C
700   RETURN
C
C   FORMAT STATEMENTS
C
1000  FORMAT(7I4)
1100  FORMAT(2I4,3F12.0)
2000  FORMAT('1',T30,'JOINT LOADS DATA AS INPUT'//,
     *'     N NMLC   NBU   NBV   NBR   INC    LC CODE',7X,'H',14X,'V',
     *14X,'R'//)
2100  FORMAT(6I5)
2200  FORMAT(30X,2I5,3D15.6)
C
      END
```

```
      SUBROUTINE LOADID(KSUBH,LCSUB,KSUB,N,NCUL,NGLC,NLC)
C
C   THIS SEGMENT PREPARES THE ARRAY KSUB FOR AN INTERMEDIATE
C   ASSEMBLY UNIT, DURING BACK SUBSTITUTION.
C
C*********************************************************************
C
      DIMENSION KSUBH(NGLC,1),LCSUB(NLC,1),KSUB(NGLC,1)
C
      DO 10 J=1,NGLC
      DO 10 I=1,NCUL
C
      KSUB(J,I)  = LCSUB(KSUBH(J,N),I)
C
10    CONTINUE
C
      RETURN
C
      END
```

```
      SUBROUTINE MLOADS(X,Y,FEF,B,NOD,MKODE,IDRY,UNIT,IN,IO,NLC,
     *NJ)
C
C   THIS SUBROUTINE  READS MEMBER LOADS AND CALCULATES THE END
C   FORCES. THE END FORCES ARE STORED IN VECTOR B AS EQUIVALENT
C   LOADS, AND IN VECTOR FEF AS MEMBER END FORCES. THE JOINT
C   LOADS ARE THEN ADDED TO B.
C
C*******************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION X(1),Y(1),FEF(NLC,1),B(NLC,1),NOD(2,1),MKODE(1),
     *FQ(6),FK(6)
C
C   MODIFY UNITS OF LOAD VECTOR
C
      DO 200 J=1,NLC
      DO 200 I=1,NJ
      II = 3*I
      B(J,II)    = B(J,II)*UNIT
200   CONTINUE
C
C   READ AND PREPARE MEMBER LOADS
C
      C13 = 1./3.
      WRITE(IO,2000)
250   READ(IN,1000) M,LC,INC,K,CL,CN,QI,AI,QJ,AJ
      IF(M.LE.0) GO TO 800
      KKK = 0
      MM = M
275   IF(IDRY.EQ.1) GO TO 700
      I = NOD(1,MM)
      J = NOD(2,MM)
      DX = (X(J) - X(I))
      DY = (Y(J) - Y(I))
      XL = DSQRT(DX**2+DY**2)
      CT = DX/XL
      ST = DY/XL
      DC     =   1./DSQRT(CL**2+CN**2)
      COSG   = DC*CL
      SING   = DC*CN
      AIM    = 1. - AI
C
      GO TO (300,400,500),K
C
C   FIXED END FORCES FOR CONCENTRATED FORCES
C
300   COSG = COSG*QI
      SING = SING*QI
      FQ(3)  = -AIM**2*AI*XL*SING
      FQ(6)  =  AIM*AI**2*XL*SING
      DV     = (FQ(3) + FQ(6))/XL
      FQ(1)  = -AIM*COSG
      FQ(2)  = -AIM*SING + DV
      FQ(4)  = -AI*COSG
      FQ(5)  = -AI*SING - DV
      GO TO 600
C
C   FIXED END FORCES FOR UNIFORM LOADS
C
400   D1 = AJ - AI
      D2 = 0.5*(AJ**2 - AI**2)
      D3 = C13*(AJ**3 - AI**3)
      D4 = 0.25*(AJ**4 - AI**4)
      COSG   = COSG*QI*XL
      SING   = SING*QI*XL
      FQ(3)  = -XL*SING*(D2 - 2.*D3 + D4)
      FQ(6)  =  XL*SING*(D3 - D4)
      DV     =   (FQ(3) + FQ(6))/XL
      FQ(1)  = -COSG*(D1 - D2)
      FQ(2)  = -SING*(D1 - D2)  + DV
      FQ(4)  = -COSG*D2
      FQ(5)  = -SING*D2 - DV
      GO TO 600
C
C   FIXED END FORCES FOR TRAPEZOIDAL LOADS
C
500   AL     = XL/(AJ - AI)
      COSGI = COSG*QI*AL
      COSGJ = COSG*QJ*AL
      SINGI = SING*QI*AL
      SINGJ = SING*QJ*AL
      D1 = AJ - AI
      D2 =  0.5*(AJ**2 - AI**2)
      D3 = C13*(AJ**3 - AI**3)
      D4 = 0.25*(AJ**4 - AI**4)
      D5 =  0.2*(AJ**5 - AI**5)
      D6 = D2 - 2.*D3 + D4
      D7 = D3 - 2.*D4 + D5
      AII = AJ*D1 - (1.+AJ)*D2 + D3
      BII = AJ*D6 - D7
      AIJ = D2 - D3 - AI*(D1-D2)
      BIJ = D7 - AI*D6
      AJI = AJ*D2 - D3
      BJI = AJ*(D3-D4) - (D4-D5)
      AJJ = D3 - AI*D2
      BJJ = D4 -D5 - AI*(D3-D4)
C
      FQ(3)  = -XL*(SINGI*BII + SINGJ*BIJ)
      FQ(6)  =  XL*(SINGI*BJI + SINGJ*BJJ)
      DV    = (FQ(3) + FQ(6))/XL
      FQ(1)  = -(COSGI*AII + COSGJ*AIJ)
      FQ(2)  = -(SINGI*AII + SINGJ*AIJ)  + DV
      FQ(4)  = -(COSGI*AJI + COSGJ*AJJ)
      FQ(5)  = -(SINGI*AJI + SINGJ*AJJ)  - DV
C
C   MODIFY END FORCES VECTOR FQ,FOR HINGED ENDS
C
600   IF(IDRY.EQ.1) GO TO 700
      KI = MKODE(MM) + 1
```

```
      DO 605 KK = 1,6
 605  FK(KK) = FQ(KK)
      GO TO (650,620,630,610),KI
 610  FK(3) = 0.0
      FK(6) = 0.0
      GO TO 640
 620  FK(6) = FK(6) - 0.5*FK(3)
      DV   = 1.5*FK(3)/XL
      FK(3) = 0.0
      GO TO 640
 630  FK(3) = FK(3) - 0.5*FK(6)
      DV   = 1.5*FK(6)/ XL
 640  FK(2) = FK(2) - DV
      FK(5) = FK(5) + DV
 650  FK(3) = FK(3)*UNIT
      FK(6) = FK(6)*UNIT
C
C   ASSEMBLE FQ INTO FEF
C
      KEF = 6*MM - 6
      DO 675 KK=1,6
      KEF = KEF + 1
      FEF(LC,KEF) = FEF(LC,KEF) + FK(KK)
 675  CONTINUE
C
C   ASSEMBLE FQ  INTO  B(LOCAL LOAD VECTOR).
C
      II = 3*NOD(1,MM)
      B(LC,II-2) = B(LC,II-2) - FK(1)*CT + FK(2)*ST
      B(LC,II-1) = B(LC,II-1) - FK(1)*ST - FK(2)*CT
      B(LC,II)   = B(LC,II) - FK(3)
      JJ = 3*NOD(2,MM)
      B(LC,JJ-2) = B(LC,JJ-2) - FK(4)*CT + FK(5)*ST
      B(LC,JJ-1) = B(LC,JJ-1) - FK(4)*ST - FK(5)*CT
      B(LC,JJ)   = B(LC,JJ) - FK(6)
C
 700  IF(INC.EQ.0) GO TO 750
      MOLD = MOLD + IABS(INC)
      MM = MOLD
      IF(MM.GT.M) GO TO 999
      IF(MM.EQ.M) GO TO 750
      IF(INC.GT.0) GO TO 275
      GO TO 600
 750  WRITE(IO,2200) M,LC,INC,K,CL,CN,QI,AI,QJ,AJ
      MOLD = MM
      GO TO 250
C
 800  RETURN
C
 999  WRITE(IO,3000) MM,M
      STOP
C
C   FORMAT STATEMENTS
C
 1000 FORMAT(4I4,2F12.0,4F10.0)
 2000 FORMAT('1',T30,'MEMBER LOADS AS INPUT'//
     *,4X,'M    LC  INC CODE',6X,'CL',13X,'CN',13X,'QI',
     *13X,'AI',13X,'QJ',13X,'AJ'//)
 2200 FORMAT(4I5,6D15.6)
 3000 FORMAT('GENERATION INCREMENT ERROR  MM=',I4,'    M=',I4)
C
      END
```

```
      SUBROUTINE RESUB(B,BA,ORINT,BP,NPG,KSUB,IG,J,NGLC,MXND,
     *NFIBN,NIBNS,NLC)
C
C   THIS SEGMENT SUBSTITUTES THE INTER-BOUNDARY PARTITION OF A
C   HIGHER LEVEL SOLUTION VECTOR INTO A LOWER LEVEL DUMMY VECTOR
C   BA, WITH THE APPROPRIATE TRNSFORMATIONS. IT IDENTIFIES THE
C   CORRESPONDING LOCAL CASES OF LOADING FOR THIS PARTICULAR
C   LOWER UNIT, AND SUBSTITUTES RI* COMPONENTS IN THEIR PROPER
C   PLACE IN VECTOR BA. BA IS NOW READY FOR BACKSUBSTITUTION.
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION B(NLC,1),BA(NGLC,1),ORINT(1),BP(NGLC,1),NPG(MXND,1),
     *KSUB(NGLC,1)
C
      THETA= ORINT(IG)*3.14159265/180.
      S     = DSIN(THETA)
      C     = DCOS(THETA)
C
      DO 200 I=1,NGLC
      NL = 3*(NFIBN)
      DO 100 II=1,NIBNS
      NG = NPG(II,IG)
      NG = 3*NG
      BA(I,NL-2)  = BP(I,NG-2)*C + BP(I,NG-1)*S
      BA(I,NL-1)  =-BP(I,NG-2)*S + BP(I,NG-1)*C
      BA(I,NL)    = BP(I,NG)
100   NL = NL + 3
200   CONTINUE
C
      IF(NFIBN.EQ.1) GO TO 400
      NFIBN1 = NFIBN - 1
      DO 300 I=1,NGLC
      LC = KSUB(I,J)
      DO 300 II=1,NFIBN1
      NL = 3*II
      BA(I,NL-2)  = B(LC,NL-2)
      BA(I,NL-1)  = B(LC,NL-1)
      BA(I,NL)    = B(LC,NL)
300   CONTINUE
C
400   RETURN
C
      END
```

```
      SUBROUTINE STIFF(NOD,MKODE,MAXA,X,Y,AREA,RI,YMOD,A,UNT,NE)
C
C   THIS SUBROUTINE FORMS THE STIFFNESS MATRIX OF A 6 DOF PLANE
C   FRAME MEMBER. THE UPPER TRINGLE IS STORED COLUMN-WISE IN A
C   VECTOR S. IT THEN ASSEMBLES THE MEMBER STIFFNESS INTO THE
C   SUBSTRUCTURE STIFFNESS MATRIX A.
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION NOD(2,1),MKODE(1),MAXA(1),X(1),Y(1),AREA(1),
     *RI(1),YMOD(1),A(1),S(21),LM(6)
C
      DO 700 M=1,NE
      I= NOD(1,M)
      J = NOD(2,M)
      DX = (X(J) - X(I))*UNT
      DY = (Y(J) - Y(I))*UNT
      XL = DSQRT(DX**2 + DY**2)
      XLI = 1./XL
      ALP  = YMOD(M)*RI(M)*XLI
      BETA = YMOD(M)*AREA(M)*XLI
      CC   = DX*XLI
      SI   = DY*XLI
      CL   = CC*XLI
      SL   = SI*XLI
      CL2  = CL**2
      SL2  = SL**2
      CSL  = CL*SL
      C2   = CC**2
      S2   = SI**2
      SC   = CC*SI
C
      K = MKODE(M) + 1
      GO TO (150,200,250,100),K
C
C   MEMBER IS HINGED AT BOTH ENDS.
C
100   A1 = 0.0
      A2 = BETA
      A3 = 0.0
      A4 = 0.0
      A5 = 0.0
      A6 = 0.0
      A7 = 0.0
      GO TO 300
C
C   MEMBER IS HCCNTINEOUS AT BOTH ENDS.
C
150   A1 = 12.0*ALP
      A2 = BETA
      A3 = 6.0*ALP
      A4 = 4.*ALP
      A5 = A3
      A6 = A4
```

```
      A7 = 2.0*ALP
      GO TO 300
C
C   MEMBER IS HINGED AT END I ONLY.
C
200   A1 = 3.0*ALP
      A2 = BETA
      A3 = 0.0
      A4 = 0.0
      A5 = A1
      A6 = A1
      A7 = 0.0
      GO TO 300
C
C   MEMBER IS HINGED AT END J ONLY.
C
250   A1 = 3.0*ALP
      A2 = BETA
      A3 = A1
      A4 = A1
      A5 = 0.0
      A6 = 0.0
      A7 = 0.0
C
C   FORM STIFFNESS MATRIX
C
300   S(1) =  A1*SL2 + A2*C2
      S(2) = -A1*CSL + A2*CS
      S(3) = -A3*SL
      S(4) = -S(1)
      S(5) = -S(2)
      S(6) = -A5*SL
      S(7) =  A1*CL2 + A2*S2
      S(8) =  A3*CL
      S(9) = -S(2)
      S(10)= -S(7)
      S(11)=  A5*CL
      S(12)=  A4
      S(13)= -S(3)
      S(14)= -S(8)
      S(15)=  A7
      S(16)=  S(1)
      S(17)=  S(2)
      S(18)= -S(6)
      S(19)=  S(7)
      S(20)= -S(11)
      S(21)=  A6
C
      LM(3) = 3*I
      LM(2) = LM(3) - 1
      LM(1) = LM(3) - 2
      LM(6) = 3*J
      LM(5) = LM(6) - 1
      LM(4) = LM(6) - 2
C
C   ADD STIFFNESS MATRIX TO SUBSTRUCRUTE STIFFNESS MATRIX
C
      NDL = 0
      DO 600 L=1,6
      LL = LM(L)
      ML = MAXA(LL)
      KS = L
      DO 500 N= 1,6
      NN = LM(N)
      LN = LL - NN
      IF(LN) 500,400,400
400   KK = ML + LN
      KSS = KS
      IF(N.GE.L) KSS = N+NDL
      A(KK) = A(KK) + S(KSS)
500   KS  = KS  + 6 - N
600   NDL = NDL + 6 - L
C
700   CONTINUE
C
      RETURN
C
      END
```

```
      SUBROUTINE STRESS(X,Y,AREA,RI,YMOD,B,FEF2,NOD,MKODE,NE,K
     *,NGLC,UNIT,IN,IO)
C
C   THIS SUBROUTINE COMPUTES AND OUTPUTS MEMBER END FORCES FOR A
C   PLANE FRAME PROBLEM.
C
C*****************************************************************
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION X(1),Y(1),AREA(1),RI(1),YMOD(1),B(NGLC,1),FEF2(1
     *),NOD(2,1),MKODE(1)
C
      DO 500 M=1,NE
      I  = NOD(1,M)
      J  = NOD(2,M)
      DX = (X(J)  -X(I))*UNIT
      DY = (Y(J) - Y(I))*UNIT
      XLI  = 1./DSQRT(DX**2+DY**2)
      COST = DX*XLI
      SINT = DY*XLI
      C1 = 2.*YMOD(M)*RI(M)*XLI
      C2 = AREA(M)*YMOD(M)*XLI
C
      DU = B(K,(3*J-2)) - B(K,(3*I-2))
      DV = B(K,(3*J-1)) - B(K,(3*I-1))
      DP = C2*(DU*COST + DV*SINT)
      ROT= 3.*(DV*COST - DU*SINT)*XLI
      DMI= C1*(2.*B(K,(3*I)) + B(K,(3*J)) - ROT)
      DMJ= C1*(2.*B(K,(3*J)) + B(K,(3*I)) - ROT)
      DV = (DMI+DMJ)*XLI
C
C   MODIFY END FORCES FOR MEMBER TYPE
C
      KK  = MKODE(M)
      IF(KK.EQ.0) GO TO 400
      GO TO (100,200,300),KK
100   DMJ = DMJ - DMI*0.5
      DV  = DV  - 1.5*DMI*XLI
      DMI = 0.0
      GO TO 400
C
200   DMI = DMI - 0.5*DMJ
      DV  = DV  - 1.5*DMJ*XLI
      DMJ = 0.0
      GO TO 400
C
300   DV  = DV - (DMI + DMJ)*XLI
      DMI = 0.0
      DMJ = 0.0
C
400   MM = 6*M
      FEF2(MM-5) = FEF2(MM-5) - DP
      FEF2(MM-4) = FEF2(MM-4) + DV
      FEF2(MM-3) = (FEF2(MM-3) + DMI)/UNIT
      FEF2(MM-2) = FEF2(MM-2) + DP
      FEF2(MM-1) = FEF2(MM-1) - DV
      FEF2(MM)   = (FEF2(MM)   + DMJ)/UNIT
500   CONTINUE
C
      WRITE(IC,1000)
      DO 600 M=1,NE
      M2 = 6*M
      M1 = M2-5
      WRITE(IC,1100) M,NOD(1,M),NOD(2,M),(FEF2(I),I=M1,M2)
600   CONTINUE
C
      RETURN
C
C   FORMAT STATEMENTS
C
1000  FORMAT('1','MEMBER END FORCES',//,'MEM',6X,'I',4X,'J',7X,
     *'MI',13X,'VI',13X,'MI',13X,'NJ',13X,'VJ',13X,'MJ'//)
1100  FORMAT(3I5,6D15.6)
C
      END
```

APPENDIX D

EXAMPLE INPUT

D.1 INPUT DATA FOR EXAMPLE 2 ON MUSAPF.

```
  15   3  16   4   1  15   3   2           12.0    0
  16  16  16  16  16  16  16  16  16  16  16  16  16  16  16
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
   2   2   2   2   2   3   3   3   3   3   1   1   1   1   1
  11  12  13  14  15   6   7   8   9  10   1   2   3   4   5
   1  30  44   2   0  16  15
   1         0.0          24.0   0
   5        80.0          24.0   1
   6         0.0          12.0   0
  14        80.0          12.0   2
   7         0.0          36.0   0
  15        80.0          36.0   2
  16       100.0          24.0
  27       100.0          12.0
  28       100.0          36.0
  17         0.0           0.0
  25        80.0           0.0   2
  18         0.0          48.0
  26        80.0          48.0   2
  29       100.0           0.0
  30       100.0          48.0   0
      4500.0      21094.0        3.6E6
   1  18  20
  13  24  26   0   4   2   2
   2   7   9
  14  13  15   0   4   2   2
   3   1   2
  15   4   5   0   4   1   1
   4   6   8
  16  12  14   0   4   2   2
  17  26  30
  18  15  28
  19   5  16
  20  14  27
  21   7  18   0   0   0   0        1.2E3        4.E4
  37  15  26   0   4   2   2       1552.0     68629.0
  41  28  30   0   0   0   0        1.2E3        4.E4
  22   1   7                        1.2E3        4.E4
  38   5  15   0   4   1   2       1552.0     68629.0
  42  16  28                        1.2E3        4.E4
  23   6   1                        1.2E3        4.E4
  39  14   5   0   4   2   1       1552.0     68629.0
  43  27  16                        1.2E3        4.E4
  24  17   6                        1.2E3        4.E4
  40  25  14   0   4   2   2       1552.0     68629.0
  44  29  27                        1.2E3        4.E4
   1
   1   0
  18   2
   1 100         1.2E4
   2 100         0.6E4
   7   2
   1 100         1.2E4
   2 100         1.2E4
   1   2
   1 100         1.2E4
   2 100         1.2E4
   6   2
   1 100         1.2E4
   2 100          1.2E4
   0
   2   8   4   1   0   1   8
   1          0.0         48.0
   7          0.0         12.0   2
   2          0.5         48.0
   8          0.5         12.0   2
        5.E4         1.0         3.6E6
   1   1   2   3
   4   7   8   3   1   2   2
   1
   0   0
   3  33  44   2   0  21  13
   1        100.0         24.0   0
   3         60.0         24.0   1
   5         40.0         24.0   1
   6         20.0         24.0   0
   7        100.0         12.0   0
  11         60.0         12.0   2
  15         40.0         12.0   2
  17         20.0         12.0   0
   8        100.0         36.0   0
  12         60.0         36.0   2
  16         40.0         36.0   2
  18         20.0         36.0   0
  22        100.0          0.0   0
  24         80.0
  28         20.0          0.0   2
  32
  30          0.0         12.0   0
  21          0.0         24.0
  31          0.0         36.0
  23        100.0         48.0
  25         80.0         48.0
  29         20.0         48.0   2
  19         60.0         48.0
  20         40.0         48.0
  33          0.0         48.0
      1.8E3       8438.0         3.6E6
   1  33  29
   2  31  18
   3  21   6
   4  30  17
   5  29  20
   6  18  16
   7   6   5
   8  17  15
   9  20  27   0   0   0   0       1.8E5     5.9066E6
  10  16  14   0   0   0   0       1.8E5     5.9066E6
  11   5   4   0   0   0   0       1.8E5     5.9066E6
  12  15  13   0   0   0   0       1.8E5     5.9066E6
  13  27  19   0   0   0   0       1.8E5     5.9066E6
  14  14  12   0   0   0   0       1.8E5     5.9066E6
  22  10   8   0   4  -2  -2
  15   4   3   0   0   0   0       1.8E5     5.9066E6
  23   2   1   0   4  -1  -1
  16  13  11   0   0   0   0       1.8E5     5.9066E6
  24   9   7   0   4  -2  -2
  17  19  25
  21  25  23
```

```
25  31  33   0   0   0   0       400.0    13333.33
26  21  31   0   0   0   0       400.0    13333.33
27  30  21   0   0   0   0       400.0    13333.33
28  32  30   0   0   0   0       400.0    13333.33
29  18  29   0   0   0   0       576.0     27648.0
30   6  18   0   0   0   0       576.0     27648.0
31  17   6   0   0   0   0       576.0     27648.0
32  28  17   0   0   0   0       576.0     27648.0
33  14  27   0   0   0   0      2880.0    1.3824E7
34   4  14   0   0   0   0      2880.0    1.3824E7
35  13   4   0   0   0   0      2880.0    1.3824E7
36  26  13   0   0   0   0      2880.0    1.3824E7
37  10  25   0   0   0   0       576.0     27648.0
38   2  10   0   0   0   0       576.0     27648.0
39   9   2   0   0   0   0       576.0     27648.0
40  24   9   0   0   0   0       576.0     27648.0
41   8  23   0   0   0   0       400.0    13333.33
42   1   8   0   0   0   0       400.0    13333.33
43   7   1   0   0   0   0       400.0    13333.33
44  22   7   0   0   0   0       400.0    13333.33
 1
 1   0
33   2
 1 100      4800.0
 2 100      2400.0
31   2
 1 100      4800.0
 2 100      4800.0
21   2
 1 100      4800.0
 2 100      4800.0
30   2
 1 100      4800.0
 2 100      4800.0
 0
 4  96   1  11   0   0  15  15
 1   1   1   1   1   3   3   3   3   3   2   2   2   2   2
 1   0   0   0   0   1   0   0   0   0   1   0   0   0   0
15  15  15  15  15  13  13  13  13  13   8   8   8   8   8
14  31  48  65  82  15  32  49  66  83   6  23  40  57  74
18  35  52  69  86  28  45  62  79  96   7  24  41  58  75
 1  18  35  52  69  11  28  45  62  79  12  29  46  63  80
19  36  53  70  87  27  44  61  78  95  13  30  47  64  81
 2  19  36  53  70  10  27  44  61  78  14  31  48  65  82
20  37  54  71  88  26  43  60  77  94  15  32  49  66  83
 3  20  37  54  71   9  26  43  60  77  16  33  50  67  84
21  38  55  72  89  25  42  59  76  93  17  34  51  68  85
 4  21  38  55  72   8  25  42  59  76
22  39  56  73  90  17  34  51  68  85
 5  22  39  56  73  13  30  47  64  81
16  33  50  67  84  24  41  58  75  92
12  29  46  63  80   7  24  41  58  75
23  40  57  74  91
 6  23  40  57  74
 2   1   1   1   1   2   1   1   1   1   1   1   1   1   1
 0.0
 0.0
 1  86 111
 2  87 111
 3  88 111
 4  89 111
 5  90 111
 6  91 111
 7  92 111
 8  93 111
 9  94 111
10  95 111
11  96 111
END OF FILE
```

D.2 INPUT DATA FOR EXAMPLE 7 ON SISAPP.

```
   15   3  15  96   1   3   2  11          12.0    0
          0.0
          0.0
   15  15  15  15  15  13  13  13  13  13   8   8   8   8   8
   14  31  48  65  82  15  32  49  66  83   6  23  40  57  74
   18  35  52  69  86  28  45  62  79  96   7  24  41  58  75
    1  18  35  52  69  11  28  45  62  79  12  29  46  63  80
   19  36  53  70  87  27  44  61  78  95  13  30  47  64  81
    2  19  36  53  70  10  27  44  61  78  14  31  48  65  82
   20  37  54  71  88  26  43  60  77  94  15  32  49  66  83
    3  20  37  54  71   9  26  43  60  77  16  33  50  67  84
   21  38  55  72  89  25  42  59  76  93  17  34  51  68  85
    4  21  38  55  72   8  25  42  59  76
   22  39  56  73  90  17  34  51  68  85
    5  22  39  56  73  13  30  47  64  81
   16  33  50  67  84  24  41  58  75  92
   12  29  46  63  80   7  24  41  58  75
   23  40  57  74  91
    6  23  40  57  74
    1  86 111
    2  87 111
    3  88 111
    4  89 111
    5  90 111
    6  91 111
    7  92 111
    8  93 111
    9  94 111
   10  95 111
   11  96 111
    1  30  44   2   5   0  16  15
    1           0.0          24.0  0
    5          80.0          24.0  1
    6           0.0          12.0  0
   14          80.0          12.0  2
    7           0.0          36.0  0
   15          80.0          36.0  2
   16         100.0          24.0
   27         100.0          12.0
   28         100.0          36.0
   17           0.0           0.0
   25          80.0           0.0  2
   18           0.0          48.0
   26          80.0          48.0  2
   29         100.0           0.0
   30         100.0          48.0  0
        4500.0      21094.0        3.6E6
    1  18  20
   13  24  26   0   4   2   2
    2   7   9
   14  13  15   0   4   2   2
    3   1   2
   15   4   5   0   4   1   1
    4   6   8
   16  12  14   0   4   2   2
   17  26  30
   18  15  28
   19   5  16
   20  14  27
   21   7  18   0   0   0   0      1.2E3     4.E4
   37  15  26   0   4   2   2     1552.0  68629.0
   41  28  30   0   0   0   0      1.2E3     4.E4
   22   1   7                     1.2E3     4.E4
   38   5  15   0   4   1   2     1552.0  68629.0
   42  16  28                     1.2E3     4.E4
   23   6   1                     1.2E3     4.E4
   39  14   5   0   4   2   1     1552.0  68629.0
   43  27  16                     1.2E3     4.E4
   24  17   6                     1.2E3     4.E4
   40  25  14   0   4   2   2     1552.0  68629.0
   44  29  27                     1.2E3     4.E4
    1
    0   1   2   3   4   5
    1   2   1   1   1   1
   18   2
    1 100       1.2E4
    2 100       0.6E4
    7   2
    1 100       1.2E4
    2 100       1.2E4
    1   2
    1 100       1.2E4
    2 100       1.2E4
    6   2
    1 100       1.2E4
    2 100       1.2E4
    0
    0
    2   8   4   1   5   0   1   8
    1           0.0        48.0
    7           0.0        12.0   2
    2           0.5        48.0
    8           0.5        12.0   2
       5.E4           1.0          3.6E6
    1   1   2   3
    4   7   8   3   1   2   2
    1
    0  11  12  13  14  15
    1   1   1   1   1   1
    0
    0
    3  33  44   2   5   0  21  13
    1         100.0        24.0   0
    3          60.0        24.0   1
    5          40.0        24.0   1
    6          20.0        24.0   0
    7         100.0        12.0   0
   11          60.0        12.0   2
   15          40.0        12.0   2
   17          20.0        12.0   0
    8         100.0        36.0   0
   12          60.0        36.0   2
   16          40.0        36.0   2
   18          20.0        36.0   0
   22         100.0         0.0   0
   24          80.0
   28          20.0         0.0   2
   32
   3[illegible]                 12.0
```

```
    21         0.0         24.0
    31         0.0         36.0
    23       100.0         48.0
    25        80.0         48.0
    29        20.0         48.0   2
    19        60.0         48.0
    20        40.0         48.0
    33         0.0         48.0
         1.8E3        8438.0        3.6E6
     1  33  29
     2  31  18
     3  21   6
     4  30  17
     5  29  20
     6  18  16
     7   6   5
     8  17  15
     9  20  27   0   0   0   0      1.8E5     5.9066E6
    10  16  14   0   0   0   0      1.8E5     5.9066E6
    11   5   4   0   0   0   0      1.8E5     5.9066E6
    12  15  13   0   0   0   0      1.8E5     5.9066E6
    13  27  19   0   0   0   0      1.8E5     5.9066E6
    14  14  12   0   0   0   0      1.8E5     5.9066E6
    22  10   8   0   4  -2  -2
    15   4   3   0   0   0   0      1.8E5     5.9066E6
    23   2   1   0   4  -1  -1
    16  13  11   0   0   0   0      1.8E5     5.9066E6
    24   9   7   0   4  -2  -2
    17  19  25
    21  25  23
    25  31  33   0   0   0   0      400.0     13333.33
    26  21  31   0   0   0   0      400.0     13333.33
    27  30  21   0   0   0   0      400.0     13333.33
    28  32  30   0   0   0   0      400.0     13333.33
    29  18  29   0   0   0   0      576.0      27648.0
    30   6  18   0   0   0   0      576.0      27648.0
    31  17   6   0   0   0   0      576.0      27648.0
    32  28  17   0   0   0   0      576.0      27648.0
    33  14  27   0   0   0   0     2880.0     1.3824E7
    34   4  14   0   0   0   0     2880.0     1.3824E7
    35  13   4   0   0   0   0     2880.0     1.3824E7
    36  26  13   0   0   0   0     2880.0     1.3824E7
    37  10  25   0   0   0   0      576.0      27648.0
    38   2  10   0   0   0   0      576.0      27648.0
    39   9   2   0   0   0   0      576.0      27648.0
    40  24   9   0   0   0   0      576.0      27648.0
    41   8  23   0   0   0   0      400.0     13333.33
    42   1   8   0   0   0   0      400.0     13333.33
    43   7   1   0   0   0   0      400.0     13333.33
    44  22   7   0   0   0   0      400.0     13333.33
     1
     0   6   7   8   9  10
     1   2   1   1   1   1
    33   2
     1 100       4800.0
     2 100       2400.0
    31   2
     1 100       4800.0
     2 100       4800.0
    21   2
     1 100       4800.0
     2 100       4800.0
    30   2
     1 100       4800.0
     2 100       4800.0
     0
     0
END OF FILE
```